ENKBC-2.1

VAX 11/730 8085
TEST PART III

EP-ENKBC-DL-2.1
FICHE 1 OF 1

JAN 1983
COPYRIGHT© 82-83
MADE IN USA

IDENTIFICATION

Product code: ENKBC VERSION 2.1

Product name: 8085 BASED MICRO-DIAGNOSTIC FOR VAX-11/730 WCS
                 AND CPU MODULES

Product date: 20-SEP-82

Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1982 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

```
DEC             DECsystem-10    DECSYSTEM-20
DECUS           MASSBUS         PDP
UNIBUS          VAX             VMS

 -------------
|d|i|g|i|t|a|l|
 -------------
```

Table of Contents

1.0 ABSTRACT 3

2.0 HARDWARE REQUIREMENTS 3

3.0 SOFTWARE REQUIREMENTS 3

4.0 PREREQUISITES 3

5.0 OPERATING INSTRUCTIONS 3
  5.1 Options 3
  5.2 Event Flags 4
  5.3 APT Setup 4

6.0 PROGRAM FUNCTIONAL DESCRIPTION 4
  6.1 Program Overview 4
  6.2 Program Size 4
  6.3 Program Run Times 4
  6.4 Run-time Dynamics 4
  6.5 Fault Detection 5
  6.6 Performance During Hardware Failures 5
  6.7 Program Applications 5
  6.8 Test Descriptions 6

7.0 MAINTENANCE HISTORY 6

1.0 ABSTRACT

This program is an 8085 based micro-diagnostic designed to test the hardware on the WCS (Writable Control Store) and DAP (Data Path) modules of the VAX-11/730 processor.

2.0 HARDWARE REQUIREMENTS

This program will run with a minimum hardware configuration of the WCS and CPU modules present.

Other hardware modules and options may be installed without affecting the diagnostic's performance.

3.0 SOFTWARE REQUIREMENTS

This diagnostic must be run under the VAX-11/730 micro monitor (ENKAA) in a standalone environment. The micro-diagnostic is written into the top 3K of 8085 RAM. The micro-diagnostic monitor also resides in 8085 ram, where console software is normally present.

4.0 PREREQUISITES

This module should be run after running modules ENKBA and ENKBB.

5.0 OPERATING INSTRUCTIONS

This program may be executed by typing DI SE ENKBC after the micro-monitor has been loaded and the MIC> prompt has been printed at the terminal. Both the micro-monitor and this program are loaded from a TU58 cassette or downline loaded through the APT-RD port. See the OVERVIEW MANUAL for more information.

5.1 Options

Not applicable

5.2 Event Flags

Not applicable

5.3 APT Setup

The micro-monitor knows when APT is present by the position of the keyswitch and the state of a line connected to the APT-RD port.

Under APT, the diagnostic will halt on error and report error information to APT via the mailbox in 8085 RAM.

The following descibes the APT parameters needed to execute this diagnostic:
TBS

6.0 PROGRAM FUNCTIONAL DESCRIPTION

6.1 Program Overview

This program is designed to detect stuck-at faults and provide a repair tool which helps isolate the failing component. A bottom up diagnostic strategy is utilized which builds on previous tests. The tests should be run in consecutive order, to gain the best isolation of a failing component.

6.2 Program Size

This program runs in 8085 RAM memory and is approximately 3K bytes long.

6.3 Program Run Times

This program will take approximately 15 seconds to run.

6.4 Run-time Dynamics

Not applicable.

6.5 Fault Detection

The diagnostic will report errors with the following header:

```
SECT  TST ERR   EXP   REC  OTHER  MSK  MODULE
```

1. SECT is the program name (ENKBC)
2. TST is the test number that failed.
3. ERR is the error number that failed.
4. EXP is the expected (correct) data.
5. REC is the received (actual) data.
6. OTHER is any other pertinent data (such as an address). This field is not always used. N/A will be printed under OTHER when not in use. The ERRORS section in the listing will describe the contents of this field when it is used.
7. MASK is the error mask used to check the result. Bits set to a 1 in this mask correspond to bits in the result that are not checked.
8. MODULE is the suspected module that caused the failure.

6.6 Performance During Hardware Failures

A power fail will cause a rebooting of the system (if a battery backup is not present). Other hardware failures will cause normal error printouts.

6.7 Program Applications

This program can be used by field service to determine which module should be replaced and to verify that the replacement eliminated the failure. It can also be used as a repair tool to help isolate a failing component by manufacturing, field service or engineering. The program is APT compatable.

Customers may use the program to verify the basic integrity of the WCS and CPU modules in the system.

6.8 Test Descriptions

See the actual test in the listing for detailed descriptions and error analyses. A brief description of the logic being tested by each test can be found in the table of contents of the listing.

7.0 MAINTENANCE HISTORY

DATE	VERSION	DESCRIPTION
8-MAR-82	1.0	Initial release.
27-AUG-82	2.0	RD Changes.
20-SEP-82	2.1	Modify TEST 1B. Fix 1A and 1B error routines to print WCS module.

```
;        ENKBC  8085 based diagnostic for WCS/DAP module  Rev 02.10
;        Table of Contents
;
;   143  DATA AREA
;   306  PROGRAM ENTRY POINT
;   324  VARIABLES AREA
;   473  PROGRAM SPECIFIC SUBROUTINES
;   474    CHECK_RD
;   493    LD_SUPPORT
;   548    LS_WRITE
;   579    LS_READ
;   609    FAST_WRITE_WRO
;   659    FAST_READ_WRO
;   686    READ_BYTE_WRO
;   718    EXEC_INST
;   734    LD_CSR_INST
;   774    LOAD_MOD_DAT
;   789    COM_ERR_BYTE_1
;   821    COM_1A_1B_ERR_1
;   855    TEST10_ERROR
;   870    TEST10_SHIFT_OS
;   915    TEST11_ERROR
;   945    TEST12_ERROR
;   960    TEST13_ERROR
;   986    TEST14_15_ERROR
;  1003    TEST16_ERROR
;  1019    TEST17_18_ERROR
;  1036    TEST19_ERROR
;  1054    TEST1A_ERROR
;  1071    TEST1B_ERROR
;  1089    TEST1B_2_3_ERROR
;  1106    TEST1C_ERROR
;  1133    TEST1D_ERROR
;  1160  TEST 10 - OS DATA TEST
;  1273  TEST 11 - LS INDEXING TEST
;  1394  TEST 12 - ALU CONDITION CODE TEST
;  1528  TEST 13 - INC WORKING REGISTER TEST
;  1592  TEST 14 - Z BIT LONG WORD TEST
;  1696  TEST 15 - 2901 XOR (WR WITH WR) Z BIT SET/CLEAR TEST
;  1895  TEST 16 - 2901 XOR (WR WITH LS) Z BIT SET/CLEAR TEST
;  2109  TEST 17 - CPU ACK SET/CLEAR
;  2207  TEST 18 - CPU ATTN SET/CLEAR
;  2306  TEST 19 - CLOCK STALL
;  2467  TEST 1A - DC LOW
;  2522  TEST 1B - AC LOW
;  2702  TEST 1C - DATA TEST ON STACK
;  2793  TEST 1D - WCS STACK BIT 14 TEST
```

```
.TITLE  "ENKBC  8085 based diagnostic for WCS/DAP module  Rev 02.10"
;
;
;                       COPYRIGHT (c) 1982 BY
;           DIGITAL EQUIPMENT CORPORATION, MAYNARD,
;             MASSACHUSETTS.  ALL RIGHTS RESERVED.
;
; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY  BE  USED  AND  COPIED
; ONLY  IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
; OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER COPIES THEREOF
; MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER  PERSON.  NO
; TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.
;
; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND
; SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
;
; DIGITAL  ASSUMES  NO  RESPONSIBILITY  FOR  THE  USE OR RELIABILITY OF ITS
; SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.
;
;
;++
; FACILITY:      VAX-11/730 MICRO-DIAGNOSTIC.
;
; ABSTRACT:      This 8085 based diagnostic tests the hardcore logic of the
;                WCS and DAP modules.
;
; ENVIRONMENT:   ENKAA  Micro-diagnostic Monitor
;
; AUTHOR:        Jim Klumpp       6-AUG-81
;
; MODIFIED BY:   David Mayo       8-MAR-82        VERSION 01.00
;
;                Ernie Preisig    27-AUG-82       VERSION 02.00   RD Changes
;
;                Mike Ricchetti   20-SEP-82       VERSION 02.10
;                Modify test 1B to check the 8085 SID line. Fix TEST 1A and
;                TEST 1B error routines to print WCS module.
;--
```

```
7000  128
7000  129
7000  130
7000  131
7000  132
7000  133
7000  134
7000  135
7000  136
7000  137
7000  138
7000  139
7000  140 000                                                   .PSECT  CPU3,BYTE
7000  141                                                       .ALIGN  BYTE
7000  142
7000  143                                       ;*****************************************************************************
7000  144                                       ;
7000  145                                       ; DATA AREA
7000  146                                       ;
7000  147                                       ;*****************************************************************************
7000  148
7000  149 00000000                              HEAD            =0
7000  150
7000  151 00000000                                              EQUATE                  ;REGISTER EQUATE MACRO
          00000002
          00000004
          00000006
          00000006
          00000001
          00000003
          00000005
          00000007 00000006
7000  152
7000  153 00000000                              UPC14A          =<^X00>                 ;(RO) UPC BIT 14 ASSERTED HIGH <MSB>
7000  154
7000  155
7000  156 00000001                              MISC2           =<^X01>
7000  157 00000001                              BOOTF           =<^X01>                 ;(RO) BOOT FLAG. ASSERTED LOW. <LSB>
7000  158
7000  159
7000  160 00000002                              DCLO            =<^X02>                 ;(RO) DC LOW FLAG ASSERTED HIGH<MSB>
7000  161
7000  162 00000002                              HLTFLG          =<^X02>                 ;(RO) HALT FLAG ASSERTED LOW   <LSB>
7000  163
7000  164
7000  165 00000003                              READJ1          =<^X03>                 ;(RO) JUMPER J1 FLAG ASS. HIGH <MSB>
7000  166
7000  167 00000003                              ALLOWP          =<^X03>                 ;(RO) ENABLE CONTROL P ASS LOW <LSB>
7000  168
7000  169
7000  170 00000004                              READJ2          =<^X04>                 ;(RO) BIT 07 EQUALS J2 <MSB>
7000  171 00000004                              APTFLG          =<^X04>                 ;(RO) APT PRESENT FLAG <LSB>
7000  172
7000  173 00000005                              AUTO            =<^X05>                 ;(RO) AUTO TEST FLAG ASS LOW <LSB>
7000  174
7000  175 00000006                              REMDIS          =<^X06>                 ;(RO) REMOTE DISABLE FLAG ASS LOW <LSB>
7000  176
7000  177 00000007                              OKV15           =<^X07>                 ;(RO) 15 VOLTS OK FLAG ASS LOW <MSB>
7000  178
7000  179 00000007                              BOOTEN          =<^X07>                 ;(RO) BOOT ENABLE FLAG ASS LOW <LSB>
```

```
7000  180
7000  181 00000008              WCSB0       =<^X08>         ;(WO) WCS WRITE DATA REGISTER
7000  182 00000009              WCSB1       =<^X09>
7000  183 0000000A              WCSB2       =<^X0A>
7000  184
7000  185 0000001C              ITDB        =<^X1C>         ;(R/W) TIMER DATA REGISTER
7000  186 0000001D              ITCSR       =<^X1D>         ;(R/W) TIMER STATUS REGISTER
7000  187
7000  188 00000020              CLRCLK      =<^X20>         ;(WO) CLEAR CPU CLOCK RUN
7000  189 00000021              SETCLK      =<^X21>         ;(WO) SET CPU CLOCK RUN
7000  190
7000  191 00000022              CLRSS       =<^X22>         ;(WO) CLEAR CLOCK SINGLE STEP
7000  192 00000023              SETSS       =<^X23>         ;(WO) SET CLOCK SINGLE STEP
7000  193
7000  194 00000024              CLRCSK      =<^X24>         ;(WO) CLEAR CSR CLOCK
7000  195 00000025              SETCSK      =<^X25>         ;(WO) SET CSR CLOCK
7000  196
7000  197 00000026              CLRUPK      =<^X26>         ;(WO) CLEAR THE UPC CLOCK
7000  198 00000027              SETUPK      =<^X27>         ;(WO) SET THE UPC CLOCK
7000  199
7000  200 00000028              SETMCI      =<^X28>         ;(WO) SET MEMORY CONTROL INIT
7000  201 00000029              CLRMCI      =<^X29>         ;(WO) CLEAR MEMORY CONTROL INIT
7000  202
7000  203 0000002A              SETMCK      =<^X2A>         ;(WO) SET MEMORY CONTROL CLOCK
7000  204 0000002B              CLRMCK      =<^X2B>         ;(WO) CLEAR MEMORY CONTROL CLOCK
7000  205
7000  206 0000002C              CLRTMR      =<^X2C>         ;(WO) STOP TIMER
7000  207 0000002D              SETTMR      =<^X2D>         ;(WO) START TIMER COUNTING
7000  208
7000  209 0000002E              CLRBSY      =<^X2E>         ;(WO) CLEAR UNIBUS BUS BUSY
7000  210 0000002F              SETBSY      =<^X2F>         ;(WO) SET UNIBUS BUS BUSY
7000  211
7000  212 00000030              SETDCL      =<^X30>         ;(WO) SET DC LOW
7000  213 00000031              CLRDCL      =<^X31>         ;(WO) CLEAR DC LOW
7000  214
7000  215 00000032              SETACL      =<^X32>         ;(WO) SET AC LOW
7000  216 00000033              CLRACL      =<^X33>         ;(WO) CLEAR AC LOW
7000  217
7000  218 00000034              INITL       =<^X34>         ;(WO) CLEAR UNIBUS INIT
7000  219 00000035              INITH       =<^X35>         ;(WO) SET UNIBUS INIT
7000  220
7000  221 00000036              CLRWCK      =<^X36>         ;(WO) CLEAR WCS BIT 00
7000  222 00000037              SETWCK      =<^X37>         ;(WO) SET WCS BIT 00
7000  223
7000  224 00000038              CLRCSR      =<^X38>         ;(WO) CLEAR WCS BIT 01
7000  225 00000039              SETCSR      =<^X39>         ;(WO) SET WCS BIT 01
7000  226
7000  227 0000003C              CLRLIT      =<^X3C>         ;(WO) CLEAR THE RUN LIGHT
7000  228 0000003D              SETLIT      =<^X3D>         ;(WO) SET THE RUN LIGHT
7000  229
7000  230 00000044              TUDB        =<^X44>         ;(R/W) TU-58 DATA BUFFER
7000  231 00000045              TUSR        =<^X45>         ;(RO)  TU-58 STATUS REGISTER
7000  232 00000046              TUMODE      =<^X46>         ;(R/W) TU-58 MODE REGISTER 1 + 2
7000  233                                                   ;FIRST ACCESS GET MR1,SECOND GETS MR2
7000  234                                                   ;READING THE CR FORCES THE INTERNAL
7000  235                                                   ;POINTER BACK TO MR1
7000  236 00000047              TUCR        =<^X47>         ;(R/W) TU-58 COMMAND REGISTER
7000  237
7000  238 00000048              TTDB        =<^X48>         ;(R/W) TERMINAL DATA BUFFER
7000  239 00000049              TTSR        =<^X49>         ;(RO)  TERMINAL STATUS REGISTER
```

```
7000  240 0000004A                    TTMODE          =<^X4A>                 ;(R/W) TERMINAL MODE REGISTER 1 + 2
7000  241                                                                     ;FIRST ACCESS GET MR1,SECOND GETS MR2
7000  242                                                                     ;READING THE CR FORCES THE INTERNAL
7000  243                                                                     ;POINTER BACK TO MR1
7000  244 0000004B                    TTCR            =<^X4B>                 ;(R/W) TERMINAL COMMAND REGISTER
7000  245
7000  246 00000080                    READ            =<^X80>                 ;(RO) CONSOLE READ REGISTER
7000  247
7000  248 00000082                    CPUACK          =<^X82>                 ;(RO) ACKNOWLEDGE FROM CPU        <LSB>
7000  249
7000  250 00000083                    CPATTN          =<^X83>                 ;(RO) ATTENTION FROM CPU          <LSB>
7000  251
7000  252 00000084                    UPC14           =<^X84>                 ;(RO) UPC BIT 14                  <LSB>
7000  253
7000  254 00000085                    CSR07           =<^X85>                 ;(RO) CSR BIT 7                   <LSB>
7000  255
7000  256 00000086                    CSR15           =<^X86>                 ;(RO) CSR BIT 15                  <LSB>
7000  257
7000  258 00000087                    CSR23           =<^X87>                 ;(RO) CSR BIT 23                  <LSB>
7000  259
7000  260 000000A0                    ENBPAR          =<^XA0>                 ;(WO) ENABLE STALL ON PARITY ERROR
7000  261 000000A1                    DISPAR          =<^XA1>                 ;(WO) DISABLE STALL ON PARITY ERROR
7000  262
7000  263 000000A2                    VSHIFT          =<^XA2>                 ;(WO) SET THE V-BUS TO SHIFT MODE
7000  264 000000A3                    VNORML          =<^XA3>                 ;(WO) SET THE V-BUS TO NORMAL MODE
7000  265
7000  266 000000A4                    SETTIM          =<^XA4>                 ;(WO) SET THE INTERVAL TIMER INTERRUPT
7000  267 000000A5                    CLRTIM          =<^XA5>                 ;(WO) CLEAR THE INTERVAL TIMER INTRPT.
7000  268
7000  269 000000A6                    ENBMEM          =<^XA6>                 ;(WO) ENABLE MEMORY REFERENCES
7000  270 000000A7                    DISMEM          =<^XA7>                 ;(WO) DISABLE MEMORY REFERENCES
7000  271
7000  272 000000A8                    SETACK          =<^XA8>                 ;(WO) SET CONSOLE ACKNOWLEDGE
7000  273 000000A9                    CLRACK          =<^XA9>                 ;(WO) CLEAR CONSOLE ACKNOWLEDGE
7000  274
7000  275                             ;NOTE:  THE CONSOLE ACKNOWLEDGE SIGNAL IS ALSO USED AS THE UPC
7000  276                             ;SERIAL INPUT.  WHEN USED THIS WAY THE SENSE IS INVERTED.  TO SIMPLIFY
7000  277                             ;MATTERS THE NEXT TWO EQUATES ARE INCLUDED
7000  278
7000  279 000000A8                    CLRUPC          =<^XA8>                 ;(WO) CLEAR THE V-BUS SERIAL INPUT
7000  280 000000A9                    SETUPC          =<^XA9>                 ;(WO) SET THE V-BUS SERIAL INPUT
7000  281
7000  282 000000AA                    SETATN          =<^XAA>                 ;(WO) SET CONSOLE ATTENTION
7000  283 000000AB                    CLRATN          =<^XAB>                 ;(WO) CLEAR CONSOLE ATTENTION
7000  284
7000  285 000000AC                    SETPFI          =<^XAC>                 ;(WO) SET POWER FAIL INTERRUPT
7000  286 000000AD                    CLRPFI          =<^XAD>                 ;(WO) CLEAR POWER FAIL INTERRUPT
7000  287
7000  288 000000AE                    SETHLT          =<^XAE>                 ;(WO) SET THE CONSOLE HALT BIT
7000  289 000000AF                    CLRHLT          =<^XAF>                 ;(WO) CLEAR THE CONSOLE HALT BIT
7000  290
7000  291 000000CC                    WRITE           =<^XCC>                 ;(WO) THE CONSOLE WRITE REGISTER
7000  292
7000  293 000000EC                    YBUSRD          =<^XEC>                 ;(RO) Y-BUS READ
7000  294
7000  295
7000  296
7000  297                             ;ADDITIONAL DATA
7000  298
7000  299 00004C22                    PARITY_VECTOR   = <^X4C22>              ;PARITY ERROR BRANCH VECTOR.
```

```
7000  300 00004C25              POWER_VECTOR    = <^X4C25>                  ;POWER FAIL BRANCH VECTOR.
7000  301 000040D9              MIPFLG          = <^X40D9>                  ;ADDRESS OF MODEM IN PROGRESS FLAG
7000  302
7000  303 00000010              HEX_10          = <^X10>                    ;HEX VALUE OF 10.
7000  304
7000  305
7000  306                       ;*************************************************************************
7000  307                       ;
7000  308                       ;  PROGRAM ENTRY POINT.... MUST BE THE FIRST INSTUCTION IN THE PROGRAM
7000  309                       ;
7000  310                       ;*************************************************************************
7000  311
7000  312 039E'C3               ENTRY:          JMP     TEST10
7003  313
7003  314                       ;*************************************************************************
7003  315                       ;
7003  316                       ;  VERSION NUMBER AND LAST TWO LETTERS OF FILE NAME.  THESE 2 WORDS CAN NOT
7003  317                       ;  BE MOVED TO ANY OTHER LOCATION WITHOUT A MICMON CHANGE.
7003  318                       ;
7003  319                       ;*************************************************************************
7003  320
7003  321 0210                  VERSION:        .WORD           <^X0210>        ;VERSION NUMBER (REV 02.10)
7005  322 43 42                 FILE_NAME:      .ASCII          'BC'            ;LAST 2 CHARS OF FILE NAME
7007  323
7007  324                       ;*************************************************************************
7007  325                       ;
7007  326                       ; VARIABLES AREA
7007  327                       ;
7007  328                       ;*************************************************************************
7007  329
7007  330 00                    LS_ADDR:        .BYTE           0               ;LS ADDRESS POINTER.
7008  331 0000                  DATA_PNT:       .WORD           0               ;POINTS TO DATA IN TABLES.
700A  332
700A  333 2F                    SUPPORT_CODE:   .BYTE           <^X2F>          ;SIX WCS MICROINSTRUCTIONS USED
700B  334 80                                    .BYTE           <^X80>          ; FOR FAST WRITES OF WRO.
700C  335 15                                    .BYTE           <^X15>
700D  336
700D  337 A0                                    .BYTE           <^XA0>          ;*
700E  338 C0                                    .BYTE           <^XC0>
700F  339 15                                    .BYTE           <^X15>
7010  340
7010  341 5B                                    .BYTE           <^X5B>          ;*
7011  342 00                                    .BYTE           <^X00>
7012  343 18                                    .BYTE           <^X18>
7013  344
7013  345 A3                                    .BYTE           <^XA3>          ;*
7014  346 D0                                    .BYTE           <^XD0>
7015  347 1E                                    .BYTE           <^X1E>
7016  348
7016  349 A3                                    .BYTE           <^XA3>          ;*
7017  350 C0                                    .BYTE           <^XC0>
7018  351 15                                    .BYTE           <^X15>
7019  352
7019  353 08                                    .BYTE           <^X08>          ;*
701A  354 00                                    .BYTE           <^X00>
701B  355 24                                    .BYTE           <^X24>
701C  356
701C  357 BE                    MOV_WRO_LSX:    .BYTE           <^XBE>          ;INSTRUCTION USED TO PASS DATA
701D  358 00                                    .BYTE           <^X00>          ; FROM WR[0] TO AN LS LOCATION
701E  359 15                                    .BYTE           <^X15>          ; SPECIFIED DURING EXECUTION.
```

```
701F   360
701F   361 36               MOV_LSX_WR0:     .BYTE           <^X36>          ;INSTRUCTION USED TO PASS DATA
7020   362 00                                .BYTE           <^X00>          ; FROM AN LS LOCATION SPECIFIED
7021   363 15                                .BYTE           <^X15>          ; DURING EXECUTION TO WR[0].
7022   364
7022   365 36               MOV_LSOS_WR:     .BYTE           <^X36>          ;INSTRUCTION TO PASS DATA FROM
7023   366 F6                                .BYTE           <^XF6>          ; THE LS LOCATION POINTED TO BY
7024   367 15                                .BYTE           <^X15>          ; THE OS REGISTER TO WR[0].
7025   368
7025   369 20               INC_WR0:         .BYTE           <^X20>          ;INSTRUCTION USED TO INCREMENT
7026   370 40                                .BYTE           <^X40>          ; WR[0].
7027   371 15                                .BYTE           <^X15>
7028   372
7028   373 20               INC_WR0_CC:      .BYTE           <^X20>          ;INSTRUCTION TO INCREMENT WR[0]
7029   374 40                                .BYTE           <^X40>          ; AND SET THE CONDITION CODES
702A   375 35                                .BYTE           <^X35>          ; APPROPRIATELY.
702B   376
702B   377 2F               CLR_WR0_CC:      .BYTE           <^X2F>          ;INSTRUCTION TO CLEAR WR[0] AND
702C   378 80                                .BYTE           <^X80>          ; SET THE CONDITION CODES
702D   379 35                                .BYTE           <^X35>          ; APPROPRIATELY.
702E   380
702E   381 2F               X_CLR_WR1:       .BYTE           <^X2F>          ;INSTRUCTION TO CLEAR WR[1].
702F   382 82                                .BYTE           <^X82>
7030   383 95                                .BYTE           <^X95>
7031   384
7031   385 A0               X_COM_WR1:       .BYTE           <^XA0>          ;INSTRUCTION USED TO COMPLEMENT
7032   386 C2                                .BYTE           <^XC2>          ; WR[1].
7033   387 95                                .BYTE           <^X95>
7034   388
7034   389 AB               XOR_WR0_WR1:     .BYTE           <^XAB>          ;INSTRUCTION TO EXCLUSIVE OR
7035   390 80                                .BYTE           <^X80>          ; THE CONTENTS OF WR0 WITH WR1
7036   391 B5                                .BYTE           <^XB5>          ; AND SET THE CC FLAGS.
7037   392
7037   393 CD               XOR_WR0_LS0:     .BYTE           <^XCD>          ;INSTRUCTION TO EXCLUSIVE OR
7038   394 00                                .BYTE           <^X00>          ; THE CONTENTS OF WR0 WITH LS0
7039   395 35                                .BYTE           <^X35>          ; AND SET THE CC FLAGS.
703A   396
703A   397 90               CLR_CPU_ACK:     .BYTE           <^X90>          ;INSTRUCTION USED TO CLEAR CPU
703B   398 C0                                .BYTE           <^XC0>          ; ACKNOWLEDGE.
703C   399 15                                .BYTE           <^X15>
703D   400
703D   401 10               SET_CPU_ACK:     .BYTE           <^X10>          ;INSTRUCTION USED TO SET CPU
703E   402 D0                                .BYTE           <^XD0>          ; ACKNOWLEDGE.
703F   403 15                                .BYTE           <^X15>
7040   404
7040   405 90               CLR_CPU_ATTN:    .BYTE           <^X90>          ;INSTRUCTION USED TO CLEAR CPU
7041   406 C0                                .BYTE           <^XC0>          ; ATTENTION.
7042   407 15                                .BYTE           <^X15>
7043   408
7043   409 10               SET_CPU_ATTN:    .BYTE           <^X10>          ;INSTRUCTION USED TO SET CPU
7044   410 E0                                .BYTE           <^XE0>          ; ATTENTION.
7045   411 15                                .BYTE           <^X15>
7046   412
7046   413 88               JUMP_WCS0:       .BYTE           <^X88>          ;JUMP TO WCS[0] INSTRUCTION.
7047   414 00                                .BYTE           <^X00>
7048   415 04                                .BYTE           <^X04>
7049   416
7049   417 88               JUMP_WCS4:       .BYTE           <^X88>          ;JUMP TO WCS[4] INSTRUCTION.
704A   418 00                                .BYTE           <^X00>
704B   419 44                                .BYTE           <^X44>
```

```
704C
704C  08                  JSR_INST:      .BYTE          <^X08>            ;INSTRUCTION USED TO JUMP
704D  00                                 .BYTE          <^X00>            ; TO A SUBROUTINE.
704E  0C                                 .BYTE          <^X0C>
704F
704F  40                  RET_INST:      .BYTE          <^X40>            ;INSTRUCTION USED TO RETURN
7050  00                                 .BYTE          <^X00>            ; FROM A SUBROUTINE.
7051  14                                 .BYTE          <^X14>
7052
7052  40                  POP_INST:      .BYTE          <^X40>            ;INSTRUCTION USED TO POP THE
7053  00                                 .BYTE          <^X00>            ; WCS STACK.
7054  12                                 .BYTE          <^X12>
7055
7055  00                  ZERO_DAT:      .BYTE          <^X00>            ;*
7056  00                  ONE_SHIFT:     .BYTE          <^X00>            ;*
7057  00                                 .BYTE          <^X00>            ;*
7058  00                                 .BYTE          <^X00>            ;*
7059  01                                 .BYTE          <^X01>            ;STANDARD DATA PATTERNS.
705A  FF                  ONE_DAT:       .BYTE          <^XFF>            ;*
705B  FF                  ZERO_SHIFT:    .BYTE          <^XFF>            ;*
705C  FF                                 .BYTE          <^XFF>            ;*
705D  FF                                 .BYTE          <^XFF>            ;*
705E  FE                                 .BYTE          <^XFE>            ;*
705F
705F  0A39'C3             TEST19_DAT:    JMP            TEST19_PAR_ERR    ;DATA LOADED INTO PARITY VECTOR
7062                                                                      ; TO RETURN FROM A PARITY TRAP.
7062
7062  0B77'C3             TEST1B_DAT:    JMP            TEST1B_TRAP       ;DATA LOADED INTO PARITY VECTOR
7065                                                                      ; TO RETURN FROM A POWER FAIL
7065                                                                      ; TRAP.
7065
7065  FF3F                TEST1D_DAT:    .WORD          <^XFF3F>          ;UPC ADDRESS WITH BITS 0-13 SET
7067  0040                               .WORD          <^X0040>          ;UPC ADDRESS WITH BIT 14 SET.
7069
7069
7069                      ;*****************************************************************************
7069                      ;
7069                      ;  SHIFT VARIABLES - DO NOT SEPARATE PAIRS
7069                      ;
7069                      ;*****************************************************************************
7069
7069
7069  02                  TEMPA_SHIFT:   .BYTE    2                       ;TEMPORARY STORAGE LOCATION FOR
706A  0000006C            TEMPA_DAT:     .BLKB    2                       ; DATA FROM VARIOUS TESTS.
706C
706C  04                  TEMPB_SHIFT:   .BYTE    4                       ;TEMPORARY STORAGE LOCATION FOR
706D  00000071            TEMPB_DAT:     .BLKB    4                       ; DATA FROM VARIOUS TESTS.
7071
7071  04                  EXP_SHIFT:     .BYTE    4                       ;LOCATION OF EXPECTED DATA FROM
7072  00000076            EXP_DAT:       .BLKB    4                       ; VARIOUS TESTS.
7076
7076  04                  REC_SHIFT:     .BYTE    4                       ;LOCATION OF RETURNED DATA FROM
7077  0000007B            REC_DAT:       .BLKB    4                       ; VARIOUS TESTS.
707B
707B                      ;*****************************************************************************
707B                      ;
707B                      ;  CHECK_RD      THIS ROUTINE CHECKS TO SEE IF DIAGNOSTIC IS UNDER RD CONTROL.
707B                      ;                IF IT IS, A CALL IS MADE TO GCVECT SO THAT THE BOOT CODE CAN
707B                      ;                KEEP TRACK OF ELAPSED TIME.
707B                      ;*****************************************************************************
```

```
707B
707B  40D9 3A                                       CHECK_RD:      LDA    MIPFLG          ;GET STATE OF MODEM IN PROGRESS
707E  B7                                                           ORA    A               ;SET CONDITION CODES
707F  C8                                                           RZ                     ;RETURN IF ZERO
7080  0000'CD                                                      CALL   GCVECT          ;ELSE GO TO BOOT CODE IF RD
7083
7083  0000'3A                                                      LDA    CNTLC_TYPED     ;SEE IF CONTROL C TYPED (UNDER RD)
7086  0F                                                           RRC
7087  0000'DC                                                      CC     SET_FOR_CONT    ;IF SO, GO TO MONITOR.
708A
708A  C9                                                           RET                    ;BACK TO TEST
708B
708B
708B                                                ;*******************************************************************************
708B                                                ;
708B                                                ;  LD_SUPPORT  THIS ROUTINE LOADS SUPPORT CODE INTO WCS LOCATIONS 0-13 TO
708B                                                ;              ENABLE FAST READS AND WRITES OF WORKING REGISTER 0 AND THUS
708B                                                ;              OF LOCAL STORE.
708B                                                ;
708B                                                ;*******************************************************************************
708B
708B
708B  00'0A'21                                      LD_SUPPORT:    LXI    H,SUPPORT_CODE  ;POINT TO FIRST SUPPORT MICROWORD.
708E  0008'22                                                      SHLD   DATA_PNT
7091
7091  00 00 21                                                     LXI    H,0             ;POINT TO WCS ADDRESS ZERO.
7094  0000'22                                                      SHLD   WCS_ADDRESS
7097
7097  C5 46 00'00'21        *****************                      DO     6               ;LOAD FIRST SIX SUPPORT MICROWORDS
      77 06 3E              *
709F                        *                                                             ; FOR WRITING DATA TO WR0.
709F  007B'CD               *                                      CALL   CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
70A2                        *
70A2  0008'2A               *                                      LHLD   DATA_PNT        ;MOVE A MICROWORD TO WCS_VALUE.
70A5  00'00'11              *                                      LXI    D,WCS_VALUE
70A8  0000'CD               *                                      CALL   MOVER_3
70AB                        *
70AB  0000'CD               *                                      CALL   WCS_WRITE       ;WRITE IT TO WCS.
70AE                        *
70AE  00'00'21              *                                      LXI    H,WCS_ADDRESS   ;POINT TO THE NEXT WCS LOCATION.
70B1  0000'CD               *                                      CALL   INC_WORD
70B4                        *
70B4  0008'2A               *                                      LHLD   DATA_PNT        ;POINT TO THE NEXT SUPPORT MICROWORD.
70B7  23                    *                                      INX    H
70B8  23                    *                                      INX    H
70B9  23                    *                                      INX    H
70BA  0008'22               *                                      SHLD   DATA_PNT
70BD                        *
70BD  35 00'00'21           *****************                      ENDDO
      70 C1 009F'C2
70C6
70C6  C5 46 00'00'21        *****************                      DO     8               ;LOAD THE NEXT 8 MICROWORDS FOR
      77 08 3E              *
70CE                        *                                                             ; READING DATA FROM WR0.
70CE  007B'CD               *                                      CALL   CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
70D1                        *
70D1                        *
70D1  00'01'21              *                                      LXI    H,X_SHIFT_R+1   ;LOAD A SHIFT RIGHT MICROWORD TO
70D4  00'00'11              *                                      LXI    D,WCS_VALUE     ; WCS_VALUE.
70D7  0000'CD               *                                      CALL   MOVER_3
```

```
70DA  537                 *
70DA  538 0000'CD         *                              CALL    WCS_WRITE        ;WRITE IT TO WCS.
70DD  539                 *
70DD  540 00'00'21        *                              LXI     H,WCS_ADDRESS    ;POINT TO THE NEXT WCS LOCATION.
70E0  541 0000'CD         *                              CALL    INC_WORD
70E3  542                 *
70E3  543 35 00'00'21     ******************             ENDDO
          70 C1 00CE'C2
70EC  544
70EC  545 C9                                             RET
70ED  546
70ED  547
70ED  548                                ;*****************************************************************************
70ED  549                                ;
70ED  550                                ;  LS_WRITE  THIS ROUTINE WRITES FOUR BYTES OF DATA FROM DATA0-DATA3 TO LOCAL
70ED  551                                ;            STORE AT THE LOCATION GIVEN BY THE CONTENTS OF THE ACCUMUALTOR.
70ED  552                                ;
70ED  553                                ;*****************************************************************************
70ED  554
70ED  555
70ED  556 0000'32                        LS_WRITE:       STA     XD_ADDRS         ;STOPE THE LS ADDRESS IN XD_ADDRS.
70F0  557
70F0  558 80 E6                                          ANI     <^X80>           ;SAVE THE MSB.
70F2  559 07                                             RLC                      ;ROTATE LEFT ONE PLACE.
70F3  560 BE F6                                          ORI     <^XBE>           ;OR IN THE MOV WR TO LS OP CODE.
70F5  561 001C'32                                        STA     MOV_WR0_LSX      ;STORE BIT 7 OF EXTENDED ADDRESS IN
70F8  562                                                                         ; BIT 16 OF INSTRUCTION.
70F8  563
70F8  564 0000'3A                                        LDA     XD_ADDRS         ;LOAD EXTENDED ADDRESS.
70FB  565 B7                                             ORA     A                ;CLEAR THE CARRY.
70FC  566 17                                             RAL                      ;SHIFT LEFT TO CLEAR LSB (B FIELD).
70FD  567 001D'32                                        STA     MOV_WR0_LSX+1    ;STORE BITS 0-6 OF EXTENDED ADDRESS
7100  568                                                                         ; IN BITS 9-15 OF INSTRUCTION.
7100  569
7100  570 0127'CD                                        CALL    FAST_WRITE_WR0   ;WRITE DATA INTO WORKING REGISTER
7103  571                                                                         ; ZERO.
7103  572
7103  573 00'1C'21                                       LXI     H,MOV_WR0_LSX    ;MOVE THE DATA FROM WR0 TO LS.
7106  574 01BA'CD                                        CALL    EXEC_INST
7109  575
7109  576 C9                                             RET
710A  577
710A  578
710A  579                                ;****************************************************************************
710A  580                                ;
710A  581                                ;  LS_READ  THIS ROUTINE READS FOUR BYTES OF DATA FROM THE LS LOCATION
710A  582                                ;           POINTED TO BY THE ACCUMULATOR TO DATA0-DATA3.
710A  583                                ;
710A  584                                ;****************************************************************************
710A  585
710A  586
710A  587 0000'32                        LS_READ:        STA     XD_ADDRS         ;STORE THE LS ADDRESS IN XD_ADDRS.
710D  588
710D  589 80 E6                                          ANI     <^X80>           ;SAVE THE MSB.
710F  590 07                                             RLC                      ;ROTATE A LEFT ONE PLACE.
7110  591 36 F6                                          ORI     <^X36>           ;OR IN THE MOV WR0 TO LS OP CODE.
7112  592 001F'32                                        STA     MOV_LSX_WR0      ;STORE BIT 7 OF EXTENDED ADDRESS IN
7115  593                                                                         ; BIT 16 OF INSTRUCTION.
7115  594
7115  595 0000'3A                                        LDA     XD_ADDRS         ;LOAD EXTENDED ADDRESS.
```

```
7118   596 B7                                       ORA     A                  ;CLEAR THE CARRY.
7119   597 17                                       RAL                        ;SHIFT LEFT TO CLEAR LSB (B FIELD).
711A   598 0020'32                                  STA     MOV_LSX_WR0+1      ;STORE BITS 0-6 OF EXTENDED ADDRESS
711D   599                                                                     ; IN BITS 9-15 OF INSTRUCTION.
711D   600
711D   601 00'1F'21                                 LXI     H,MOV_LSX_WR0      ;EXECUTE THE INSTRUCTION.
7120   602 01BA'CD                                  CALL    EXEC_INST
7123   603
7123   604 016E'CD                                  CALL    FAST_READ_WR0      ;READ WR0.
7126   605
7126   606 C9                                       RET
7127   607
7127   608
7127   609                            ;*********************************************************************************
7127   610                            ;
7127   611                            ;   FAST_WRITE_WR0  THIS ROUTINE PERFORMS A FAST WRITE OF WORKING REGISTER
7127   612                            ;                   ZERO USING SUPPORT MICROCODE IN WCS.
7127   613                            ;
7127   614                            ;*********************************************************************************
7127   615
7127   616
7127   617 00 00 21                   FAST_WRITE_WR0: LXI     H,0                ;PREPARE TO EXECUTE WCS SUPPORT CODE
712A   618 0000'22                                  SHLD    UPC_VALUE          ; STORED IN WCS LOACATIONS 0-5.
712D   619 0000'CD                                  CALL    LOAD_UPC
7130   620
7130   621 0000'CD                                  CALL    NOP_CSR            ;LOAD CSR WITH NOP INSTRUCTION.
7133   622
7133   623 23 D3                                    OUT     SETSS              ;EXECUTE WCS INSTRUCTION TO CLEAR WR0.
7135   624 22 D3                                    OUT     CLRSS
7137   625
7137   626 23 D3                                    OUT     SETSS              ;COMPLEMENT WR0.
7139   627 22 D3                                    OUT     CLRSS
713B   628
713B   629 007B'CD                                  CALL    CHECK_RD           ;CALL BOOT CODE IF UNDER RD CONTROL
713E   630
713E   631 C5 46 00'00'21   ******************      DO      32
           77 20 3E         *
7146   632                  *
7146   633 00'04'21         *                       LXI     H,DATA_SHIFT+4     ;POINT TO END OF DATA AREA.
7149   634 04 0E            *                       MVI     C,4                ;SHIFT FOUR BYTES INTO WR0.
714B   635 0000'CD          *                       CALL    SHIFTER1
714E   636                  *
714E   637 AB D3            *                       OUT     CLRATN             ;SET A ZERO TO BE READ.
7150   638                  *
7150   639 0155'D2          * ****************      IFC
7153   640 AA D3            * *                     OUT     SETATN             ;SET A ONE TO BE READ.
7155   641                  * ****************      ENDIF
7155   642                  *
7155   643 23 D3            *                       OUT     SETSS              ;EXECUTE WCS SUBROUTINE TO SHIFT
7157   644 22 D3            *                       OUT     CLRSS              ; A BIT INTO WR0.
7159   645                  *
7159   646 23 D3            *                       OUT     SETSS              ;*
715B   647 22 D3            *                       OUT     CLRSS
715D   648                  *
715D   649 23 D3            *                       OUT     SETSS              ;*
715F   650 22 D3            *                       OUT     CLRSS
7161   651                  *
7161   652 35 00'00'21      ******************      ENDDO
           70 C1 0146'C2
716A   653
```

```
716A  654 007B'CD                                    CALL    CHECK_RD          ;CALL BOOT CODE IF UNDER RD CONTROL
716D  655
716D  656 C9                                         RET
716E  657
716E  658
716E  659                           ;*****************************************************************************
716E  660                           ;
716E  661                           ;  FAST_READ_WRO  THIS ROUTINE PERFORMS A FAST READ OF WORKING REGISTER ZERO
716E  662                           ;                 USING SUPPORT MICROCODE IN WCS.
716E  663                           ;
716E  664                           ;*****************************************************************************
716E  665
716E  666
716E  667 00'00'21                  FAST_READ_WRO:   LXI     H,X_MOV_WRWR      ;ENABLE WRO TO BE READ.
7171  668 01C2'CD                                    CALL    LD_CSR_INST
7174  669
7174  670 EC D3                                      OUT     YBUSRD            ;LATCH THE DATA IN THE Y-BUS LATCH.
7176  671 80 DB                                      IN      READ              ;READ THE Y-BUS.
7178  672 0000'32                                    STA     DATA3             ;STORE IN DATA 3.
717B  673
717B  674 018E'CD                                    CALL    READ_BYTE_WRO     ;READ THE NEXT BYTE OF WRO.
717E  675 0000'32                                    STA     DATA2
7181  676
7181  677 018E'CD                                    CALL    READ_BYTE_WRO     ;AND THE NEXT...
7184  678 0000'32                                    STA     DATA1
7187  679
7187  680 018E'CD                                    CALL    READ_BYTE_WRO     ;AND THE LAST.
718A  681 0000'32                                    STA     DATA0
718D  682
718D  683 C9                                         RET
718E  684
718E  685
718E  686                           ;*****************************************************************************
718E  687                           ;
718E  688                           ;  READ_BYTE_WRO  THIS ROUTINE EXECUTES SUPPORT CODE OUT OF WCS WHICH SHIFTS
718E  689                           ;                 THE CONTENTS OF WORKING REGISTER ZERO TO THE RIGHT EIGHT
718E  690                           ;                 PLACES. IT THEN READS THE LOW BYTE OF WRO AND LEAVES IT
718E  691                           ;                 IN THE ACCUMULATOR.
718E  692                           ;
718E  693                           ;*****************************************************************************
718E  694
718E  695
718E  696 06 00 21                  READ_BYTE_WRO:   LXI     H,<^X0600>        ;PREPARE TO EXECUTE WCS SUPPORT CODE
7191  697 0000'22                                    SHLD    UPC_VALUE         ; STORED IN WCS LOACATIONS 6-D.
7194  698 0000'CD                                    CALL    LOAD_UPC
7197  699
7197  700 0000'CD                                    CALL    NOP_CSR           ;LOAD CSR WITH NOP INSTRUCTION.
719A  701
719A  702 C5 46 00'00'21  ******************          DO      9
          77 09 3E        *
71A2  703                 *
71A2  704 23 D3           *                          OUT     SETSS             ;EXECUTE A WCS MICROINSTRUCTION.
71A4  705 22 D3           *                          OUT     CLRSS
71A6  706                 *
71A6  707 35 00'00'21     ******************          ENDDO
          70 C1 01A2'C2
71AF  708
71AF  709 00'00'21                                   LXI     H,X_MOV_WRWR      ;ENABLE WRO TO BE READ.
71B2  710 01C2'CD                                    CALL    LD_CSR_INST
71B5  711
```

```
71B5  EC D3                                                   OUT     YBUSRD          ;LATCH THE DATA IN THE Y-BUS LATCH.
71B7  80 DB                                                   IN      READ            ; READ THE Y-BUS.
71B9
71B9  C9                                                      RET
71BA
71BA
71BA                                          ;**********************************************************************
71BA                                          ;
71BA                                          ;  EXEC_INST  THIS ROUTINE EXECUTES AN INSTRUCTION POINTED TO BY THE H,L
71BA                                          ;             PAIR.
71BA                                          ;
71BA                                          ;
71BA                                          ;**********************************************************************
71BA
71BA
71BA  01C2'CD                                 EXEC_INST:      CALL    LD_CSR_INST     ;LOAD INSTRUCTION INTO THE CSR.
71BD
71BD  23 D3                                                   OUT     SETSS           ;CLOCK THE CPU.
71BF  22 D3                                                   OUT     CLRSS
71C1
71C1  C9                                                      RET
71C2
71C2
71C2                                          ;**********************************************************************
71C2                                          ;
71C2                                          ;  LD_CSR_INST  THIS ROUTINE LOADS THE CSR WITH AN INSTRUCTION, WHICH IS
71C2                                          ;               POINTED TO BY THE CONTENTS OF THE H,L PAIR.
71C2                                          ;
71C2                                          ;
71C2                                          ;**********************************************************************
71C2
71C2  00'00'11                                LD_CSR_INST:    LXI     D,CSR_VALUE     ;MOVE INSTRUCTION POINTED TO BY THE
71C5  0000'CD                                                 CALL    MOVER_3         ; H,L PAIR TO CSR_VALUE.
71C8
71C8  A2 D3                                                   OUT     VSHIFT          ;SET V-BUS TO SHIFTING MODE.
71CA
71CA  00'00'21                                                LXI     H,CSR_SHIFT     ;PREPARE TO SHIFT DATA INTO CSR.
71CD  0000'22                                                 SHLD    SHIFT_PNT
71D0
71D0  C5 46 00'00'21      ******************                  DO      24
      77 18 3E            *
71D8                      *
71D8  38 D3               *                                   OUT     CLRCSR          ;CLEAR THE CSR SERIAL INPUT.
71DA                      *
71DA  87 DB               *                                   IN      CSR23           ;READ THE CSR'S MSB.
71DC  1F                  *                                   RAR                     ;ROTATE IT INTO THE CARRY.
71DD  0000'CD             *                                   CALL    SHIFTER         ;SHIFT THE DATA THROUGH CSR_VALUE.
71E0                      *
71E0  01E5'D2             *                                   JNC     1$              ;IF THE CARRY IS SET...
71E3                      *
71E3  39 D3               *                                   OUT     SETCSR          ;SET THE CSR SERIAL INPUT.
71E5                      *
71E5  25 D3               *                   1$:             OUT     SETCSK          ;CLOCK THE CSR SERIAL LOAD.
71E7  24 D3               *                                   OUT     CLRCSK
71E9                      *
71E9  35 00'00'21         ******************                  ENDDO
      70 C1 01D8'C2
71F2
71F2  A3 D3                                                   OUT     VNORML          ;SET V-BUS TO NORMAL MODE.
71F4
71F4  007B'CD                                                 CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
```

```
71F7  770
71F7  771 C9                                   RET
71F8  772
71F8  773
71F8  774                           ;*************************************************************************
71F8  775                           ;
71F8  776                           ;  LOAD_MOD_DAT  THIS ROUTINE LOADS THE MODULE DATA POINTED TO BY THE H,L
71F8  777                           ;                INTO CON_MOD_DAT.
71F8  778                           ;
71F8  779                           ;*************************************************************************
71F8  780
71F8  781
71F8  782 00'00'11                  LOAD_MOD_DAT:   LXI     D,CON_MOD_DAT    ;LOAD THE MODULE DATA POINTED TO BY
71FB  783 05 0E                                     MVI     C,5              ; THE H,L PAIR INTO CON_MOD_DAT.
71FD  784 0000'CD                                   CALL    MOVER
7200  785
7200  786 C9                                        RET
7201  787
7201  788
7201  789                           ;*************************************************************************
7201  790                           ;
7201  791                           ;  COM_ERR_BYTE_1: THIS ROUTINE IS USED BY ALL TESTS THAT REPORT ONE BYTE
7201  792                           ;                  OF DATA AS EXP AND REC DATA AND AS ERROR 1.
7201  793                           ;
7201  794                           ;  COM_ERR_BYTE: THIS ROUTINE IS USED BY ALL TESTS THAT REPORT ONE BYTE
7201  795                           ;                OF DATA AS EXP AND REC DATA.
7201  796                           ;
7201  797                           ;*************************************************************************
7201  798
7201  799
7201  800 01 3E                     COM_ERR_BYTE_1: MVI     A,1              ;ERROR NUMBER 1.
7203  801 0000'32                                   STA     CON_ERR_NUM
7206  802
7206  803 FF FF 21                  COM_ERR_BYTE:   LXI     H,<^XFFFF>       ;MASK OFF UPPER WORD WITH ERROR
7209  804 0000'22                                   SHLD    CON_MSK_DAT      ; MASK
720C  805
720C  806 0072'3A                                   LDA     EXP_DAT          ;MOVE EXPECTED DATA TO CON_EXP_DAT.
720F  807 0003'32                                   STA     CON_EXP_DAT+3
7212  808
7212  809 0077'3A                                   LDA     REC_DAT          ;MOVE RECEIVED DATA TO CON_REC_DAT.
7215  810 0003'32                                   STA     CON_REC_DAT+3
7218  811
7218  812 0000'32 3C AF            COM_ERR:        SET     NA_OTHER         ;NO ADDITIONAL DATA.
721D  813
721D  814 00'00'21                  COM_ERR_OTHER:  LXI     H,MCPU_A         ;CPU MODULE BEING TESTED.
7220  815 01F8'CD                                   CALL    LOAD_MOD_DAT
7223  816
7223  817 0000'CD                                   CALL    CON_ERROR        ;PRINT AN ERROR.
7226  818
7226  819 C9                                        RET                      ;BACK TO ERROR ROUTINE
7227  820
7227  821                           ;*************************************************************************
7227  822                           ;
7227  823                           ;  COM_1A_1B_ERR_1: THIS ROUTINE IS USED BY TESTS 1A AND 1B TO REPORT ONE BYTE
7227  824                           ;                   OF DATA AS EXP AND REC DATA AND AS ERROR 1.
7227  825                           ;
7227  826                           ;  COM_1A_1B_ERR: THIS ROUTINE IS USED BY TESTS 1A AND 1B TO REPORT ONE BYTE
7227  827                           ;                 OF DATA AS EXP AND REC DATA.
7227  828                           ;
7227  829                           ;*************************************************************************
```

```
7227
7227
7227                                        COM_1A_1B_ERR_1:
7227  01 3E                                                  MVI     A,1              ;ERROR NUMBER 1.
7229  0000'32                                                STA     CON_ERR_NUM
722C
722C  FF FF 21                              COM_1A_1B_ERR:   LXI     H,<^XFFFF>       ;MASK OFF UPPER WORD WITH ERROR
722F  0000'22                                                SHLD    CON_MSK_DAT      ; MASK
7232
7232  0072'3A                                                LDA     EXP_DAT          ;MOVE EXPECTED DATA TO CON_EXP_DAT.
7235  0003'32                                                STA     CON_EXP_DAT+3
7238
7238  0077'3A                                                LDA     REC_DAT          ;MOVE RECEIVED DATA TO CON_REC_DAT.
723B  0003'32                                                STA     CON_REC_DAT+3
723E
723E  0000'32 3C AF                                          SET     NA_OTHER         ;NO ADDITIONAL DATA.
7243
7243  00'00'21                                               LXI     H,MWCS_A         ;WCS MODULE BEING TESTED.
7246  01F8'CD                                                CALL    LOAD_MOD_DAT
7249
7249  0000'CD                                                CALL    CON_ERROR        ;PRINT AN ERROR.
724C
724C  C9                                                     RET                      ;BACK TO ERROR ROUTINE
724D
724D
724D                                        ;*****************************************************************************
724D                                        ;
724D                                        ;  TEST10_ERROR  THIS ROUTINE IS USED BY TEST 10 TO DEFINE THE ERROR VARIABLES
724D                                        ;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
724D                                        ;
724D                                        ;*****************************************************************************
724D
724D
724D  00 FF 21                              TEST10_ERROR:    LXI     H,<^X00FF>       ;ONLY THE LEAST SIGNIFICANT BYTE IS
7250  0002'22                                                SHLD    CON_MSK_DAT+2    ; OF INTEREST.
7253
7253  0201'CD                                                CALL    COM_ERR_BYTE_1   ;REPORT ERROR 1 (1 BYTE OF DATA)
7256
7256  C9                                                     RET
7257
7257                                        ;*****************************************************************************
7257                                        ;
7257                                        ;  TEST10_SHIFT_OS  THIS ROUTINE IS USED BY TEST 10 TO TEST THE OS REG WITH
7257                                        ;               A SHIFT PATTERN.
7257                                        ;
7257                                        ;*****************************************************************************
7257
7257                                        TEST10_SHIFT_OS:
7257  C5 46 00'00'21    ******************                   DO      8
      77 08 3E          *
725F                    *
725F  006D'3A           *                   3$:              LDA     TEMPB_DAT        ;LOAD THE TEST DATA INTO EXP_DAT AND
7262  0072'32           *                                    STA     EXP_DAT          ; DATA3.
7265  0000'32           *                                    STA     DATA3
7268                    *
7268  00'3E             *                                    MVI     A,OS_ADD         ;WRITE TO THE OS REGISTER VIA WRO.
726A  00ED'CD           *                                    CALL    LS_WRITE
726D                    *
726D  00'01'21          *                                    LXI     H,X_CLR_WRO+1    ;CLEAR WRO.
7270  01BA'CD           *                                    CALL    EXEC_INST
```

```
7273  889                     *
7273  890 00'3E               *                                   MVI     A,OS_ADD          ;READ THE OS REGISTER.
7275  891 010A'CD             *                                   CALL    LS_READ
7278  892                     *
7278  893 0000'3A             *                                   LDA     DATA3             ;MOVE THE RECEIVED DATA TO REC_DAT.
727B  894 0077'32             *                                   STA     REC_DAT
727E  895                     *
727E  896 00'72'21            * ****************                  IFNEQ   EXP_DAT,REC_DAT,1         ;COMPARE THE EXPECTED AND
          01 0E 00'77'11      * *
          028F'CA 0000'CD     * *
728C  897                     * *                                                                 ; RECEIVED DATA.
728C  898                     * *
728C  899 024D'CD             * *                                 CALL    TEST10_ERROR      ;PRINT AN ERROR MESSAGE.
728F  900                     * *
728F  901                     * ****************                  ENDIF
728F  902                     *
728F  903 0000'3A             *                                   LDA     ERROR_CON         ;IF THE ERROR LOOPING FLAG IS SET...
7292  904 0F                  *                                   RRC
7293  905 025F'DA             *                                   JC      3$                ;...JUMP TO BEGINNING OF ERROR LOOP.
7296  906                     *
7296  907 006D'3A             *                                   LDA     TEMPB_DAT         ;ROTATE THE ONE THROUGH THE ZERO
7299  908 07                  *                                   RLC                       ; FIELD.
729A  909 006D'32             *                                   STA     TEMPB_DAT
729D  910                     *
729D  911 35 00'00'21         ******************                  ENDDO
          70 C1 025F'C2
72A6  912
72A6  913 C9                                                      RET
72A7  914
72A7  915                                        ;*************************************************************************
72A7  916                                        ;
72A7  917                                        ;  TEST11_ERROR  THIS ROUTINE IS USED BY TEST 11 TO DEFINE THE ERROR VARIABLES
72A7  918                                        ;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
72A7  919                                        ;
72A7  920                                        ;*************************************************************************
72A7  921
72A7  922
72A7  923 01 3E                                  TEST11_ERROR:    MVI     A,1               ;ERROR NUMBER 1
72A9  924 0000'32                                                 STA     CON_ERR_NUM
72AC  925
72AC  926 00'72'21                                                LXI     H,EXP_DAT         ;MOVE EXPECTED DATA TO CON_EXP_DAT.
72AF  927 00'00'11                                                LXI     D,CON_EXP_DAT
72B2  928 0000'CD                                                 CALL    MOVER_4
72B5  929
72B5  930 00'77'21                                                LXI     H,REC_DAT         ;MOVE RECEIVED DATA TO CON_REC_DAT.
72B8  931 00'00'11                                                LXI     D,CON_REC_DAT
72BB  932 0000'CD                                                 CALL    MOVER_4
72BE  933
72BE  934 03 0E 00'00'21                                          ZERO    CON_ADD_DAT,3
          0D 23 77 AF
          02C4'C2
72CA  935 0007'3A                                                 LDA     LS_ADDR           ;STORE THE LS POINTER IN CON_ADD_DAT.
72CD  936 0003'32                                                 STA     CON_ADD_DAT+3
72D0  937
72D0  938 04 0E 00'00'21                                          ZERO    CON_MSK_DAT,4     ;ERROR MASK INCLUDES ALL BYTES.
          0D 23 77 AF
          02D6'C2
72DC  939
72DC  940 021D'CD                                                 CALL    COM_ERR_OTHER     ;SETUP AND REPORT ERROR WITH OTHER DATA
72DF  941
```

```
72DF   942 C9                                          RET
72E0   943
72E0   944
72E0   945                             ;*************************************************************************
72E0   946                             ;
72E0   947                             ;  TEST12_ERROR  THIS ROUTINE IS USED BY TEST 12 TO DEFINE THE ERROR VARIABLES
72E0   948                             ;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
72E0   949                             ;
72E0   950                             ;*************************************************************************
72E0   951
72E0   952 F0 FF 21                    TEST12_ERROR:   LXI     H,<^XF0FF>      ;ONLYLOW FOUR BITS ARE OF INTEREST
72E3   953 0002'22                                     SHLD    CON_MSK_DAT+2
72E6   954
72E6   955 0201'CD                                     CALL    COM_ERR_BYTE_1  ;REPORT ERROR 1 (1 BYTE OF DATA)
72E9   956
72E9   957 C9                                          RET
72EA   958
72EA   959
72EA   960                             ;*************************************************************************
72EA   961                             ;
72EA   962                             ;  TEST13_ERROR  THIS ROUTINE IS USED BY TEST 13 TO DEFINE THE ERROR VARIABLES
72EA   963                             ;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
72EA   964                             ;
72EA   965                             ;*************************************************************************
72EA   966
72EA   967
72EA   968 01 3E                       TEST13_ERROR:   MVI     A,1             ;ERROR NUMBER 1.
72EC   969 0000'32                                     STA     CON_ERR_NUM
72EF   970
72EF   971 00'72'21                                    LXI     H,EXP_DAT       ;MOVE EXPECTED DATA TO CON_EXP_DAT.
72F2   972 00'00'11                                    LXI     D,CON_EXP_DAT
72F5   973 0000'CD                                     CALL    MOVER_4
72F8   974
72F8   975 00'77'21                                    LXI     H,REC_DAT       ;MOVE RECEIVED DATA TO CON_REC_DAT.
72FB   976 00'00'11                                    LXI     D,CON_REC_DAT
72FE   977 0000'CD                                     CALL    MOVER_4
7301   978
7301   979 04 0E 00'00'21                              ZERO    CON_MSK_DAT,4   ;ERROR MASK INCLUDES ALL BYTES.
           0D 23 77 AF
           0307'C2
730D   980
730D   981 0218'CD                                     CALL    COM_ERR         ;SETUP AND REPORT ERROR
7310   982
7310   983 C9                                          RET
7311   984
7311   985
7311   986                             ;*************************************************************************
7311   987                             ;
7311   988                             ;  TEST14_15_ERROR  THIS ROUTINE IS USED BY TESTS 14 AND 15 TO DEFINE THE ERROR
7311   989                             ;                   VARIABLES IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
7311   990                             ;
7311   991                             ;*************************************************************************
7311   992
7311   993
7311   994                             TEST14_15_ERROR:
7311   995 FB FF 21                                    LXI     H,<^XFBFF>      ;ONLY ZERO BIT (BIT 2) IS OF INTEREST
7314   996 0002'22                                     SHLD    CON_MSK_DAT+2
7317   997
7317   998 0206'CD                                     CALL    COM_ERR_BYTE    ;REPORT ERROR (1 BYTE OF DATA)
731A   999
```

```
731A 1000 C9                            RET
731B 1001
731B 1002
731B 1003                       ;*********************************************************************
731B 1004                       ;
731B 1005                       ;  TEST16_ERROR  THIS ROUTINE IS USED BY TEST 16 TO DEFINE THE ERROR VARIABLES
731B 1006                       ;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
731B 1007                       ;
731B 1008                       ;*********************************************************************
731B 1009
731B 1010 FB FF 21              TEST16_ERROR:   LXI     H,<^XFBFF>      ;ONLY THE ZERO BIT (BIT 2) IS OF
731E 1011                                                               ; INTEREST
731E 1012 0002'22                               SHLD    CON_MSK_DAT+2
7321 1013
7321 1014 0206'CD                               CALL    COM_ERR_BYTE    ;REPORT ERROR (1 BYTE OF DATA)
7324 1015
7324 1016 C9                                    RET
7325 1017
7325 1018
7325 1019                       ;*********************************************************************
7325 1020                       ;
7325 1021                       ;  TEST17_18_ERROR  THIS ROUTINE IS USED BY TESTS 17 AND 18 TO DEFINE THE ERROR
7325 1022                       ;                   VARIABLES IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
7325 1023                       ;
7325 1024                       ;*********************************************************************
7325 1025
7325 1026
7325 1027                       TEST17_18_ERROR:
7325 1028 FE FF 21                              LXI     H,<^XFEFF>      ;ONLY LSB IS OF INTEREST
7328 1029 0002'22                               SHLD    CON_MSK_DAT+2
732B 1030
732B 1031 0206'CD                               CALL    COM_ERR_BYTE    ;REPORT ERROR (1 BYTE OF DATA)
732E 1032
732E 1033 C9                                    RET
732F 1034
732F 1035
732F 1036                       ;*********************************************************************
732F 1037                       ;
732F 1038                       ;  TEST19_ERROR     THIS ROUTINE IS USED BY TEST 19 TO DEFINE THE ERROR
732F 1039                       ;                   VARIABLES IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
732F 1040                       ;
732F 1041                       ;*********************************************************************
732F 1042
732F 1043
732F 1044                       TEST19_ERROR:
732F 1045 00 FF 21                              LXI     H,<^X00FF>      ;ONLY LSB IS OF INTEREST
7332 1046 0002'22                               SHLD    CON_MSK_DAT+2
7335 1047
7335 1048 0201'CD                               CALL    COM_ERR_BYTE_1  ;REPORT ERROR 1 (1 BYTE OF DATA)
7338 1049
7338 1050 C9                                    RET
7339 1051
7339 1052
7339 1053
7339 1054                       ;*********************************************************************
7339 1055                       ;
7339 1056                       ;  TEST1A_ERROR     THIS ROUTINE IS USED BY TEST 1A TO DEFINE THE ERROR
7339 1057                       ;                   VARIABLES IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
7339 1058                       ;
7339 1059                       ;*********************************************************************
```

```
7339 1060
7339 1061
7339 1062                                 TEST1A_ERROR:
7339 1063 00 FF 21                                        LXI     H,<^X00FF>      ;ONLY LSB IS OF INTEREST
733C 1064 0002'22                                         SHLD    CON_MSK_DAT+2
733F 1065
733F 1066 0227'CD                                         CALL    COM_1A_1B_ERR_1 ;REPORT ERROR 1 (1 BYTE OF DATA)
7342 1067
7342 1068 C9                                              RET
7343 1069
7343 1070
7343 1071                                 ;*****************************************************************************
7343 1072                                 ;
7343 1073                                 ;  TEST1B_ERROR  THIS ROUTINE IS USED BY TEST 1B TO DEFINE THE ERROR VARIABLES
7343 1074                                 ;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
7343 1075                                 ;
7343 1076                                 ;*****************************************************************************
7343 1077
7343 1078
7343 1079 0000'32 3C AF                  TEST1B_ERROR:   SET     NA_EXP_REC      ;NO EXPECTED OR RECEIVED DATA.
7348 1080
7348 1081 FF FF 21                                        LXI     H,<^XFFFF>      ;ERROR MASK INCLUDES NO DATA.
734B 1082 0002'22                                         SHLD    CON_MSK_DAT+2
734E 1083
734E 1084 0227'CD                                         CALL    COM_1A_1B_ERR_1 ;REPORT ERROR 1 (NA EXP AND REC)
7351 1085
7351 1086 C9                                              RET
7352 1087
7352 1088
7352 1089                                 ;*****************************************************************************
7352 1090                                 ;
7352 1091                                 ;  TEST1B_2_3_ERROR  THIS ROUTINE IS USED BY TEST 1B TO DEFINE THE ERROR
7352 1092                                 ;                    VARIABLES IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
7352 1093                                 ;
7352 1094                                 ;*****************************************************************************
7352 1095
7352 1096
7352 1097                                 TEST1B_2_3_ERROR:
7352 1098 7F FF 21                                        LXI     H,<^X7FFF>      ;BIT 7 OF A REGISTER IS OF INTEREST.
7355 1099 0002'22                                         SHLD    CON_MSK_DAT+2
7358 1100
7358 1101 022C'CD                                         CALL    COM_1A_1B_ERR   ;REPORT ERROR 2 OR 3.
735B 1102
735B 1103 C9                                              RET
735C 1104
735C 1105
735C 1106                                 ;*****************************************************************************
735C 1107                                 ;
735C 1108                                 ;  TEST1C_ERROR  THIS ROUTINE IS USED BY TEST 1C TO DEFINE THE ERROR VARIABLES
735C 1109                                 ;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
735C 1110                                 ;
735C 1111                                 ;*****************************************************************************
735C 1112
735C 1113
735C 1114 01 3E                          TEST1C_ERROR:   MVI     A,1             ;ERROR NUMBER 1.
735E 1115 0000'32                                         STA     CON_ERR_NUM
7361 1116
7361 1117 0072'2A                                         LHLD    EXP_DAT         ;MOVE EXPECTED DATA TO CON_EXP_DAT.
7364 1118 0002'22                                         SHLD    CON_EXP_DAT+2
7367 1119
```

```
7367 1120 0077'2A                                 LHLD    REC_DAT         ;MOVE RECEIVED DATA TO CON_REC_DAT.
736A 1121 0002'22                                 SHLD    CON_REC_DAT+2
736D 1122
736D 1123 FF FF 21                                LXI     H,<^XFFFF>
7370 1124 0000'22                                 SHLD    CON_MSK_DAT
7373 1125 00 00 21                                LXI     H,0             ;SECOND TWO BYTES ARE OF INTEREST
7376 1126 0002'22                                 SHLD    CON_MSK_DAT+2
7379 1127
7379 1128 0218'CD                                 CALL    COM_ERR         ;SETUP AND REPORT ERROR
737C 1129
737C 1130 C9                                      RET
737D 1131
737D 1132
737D 1133                 ;*****************************************************************************
737D 1134                 ;
737D 1135                 ;  TEST1D_ERROR  THIS ROUTINE IS USED BY TEST 1D TO DEFINE THE ERROR VARIABLES
737D 1136                 ;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
737D 1137                 ;
737D 1138                 ;*****************************************************************************
737D 1139
737D 1140
737D 1141 01 3E                 TEST1D_ERROR:     MVI     A,1             ;ERROR NUMBER 1.
737F 1142 0000'32                                 STA     CON_ERR_NUM
7382 1143
7382 1144 0072'2A                                 LHLD    EXP_DAT         ;MOVE EXPECTED DATA TO CON_EXP_DAT.
7385 1145 0002'22                                 SHLD    CON_EXP_DAT+2
7388 1146
7388 1147 0077'2A                                 LHLD    REC_DAT         ;MOVE RECEIVED DATA TO CON_REC_DAT.
738B 1148 0002'22                                 SHLD    CON_REC_DAT+2
738E 1149
738E 1150 FF FF 21                                LXI     H,<^XFFFF>      ;ONLY BIT 14 OF THE UPC ADDRESS
7391 1151 0000'22                                 SHLD    CON_MSK_DAT     ; IS OF INTEREST.
7394 1152 FF BF 21                                LXI     H,<^XFFBF>
7397 1153 0002'22                                 SHLD    CON_MSK_DAT+2
739A 1154
739A 1155 0218'CD                                 CALL    COM_ERR         ;SETUP AND REPORT ERROR
739D 1156
739D 1157 C9                                      RET
739E 1158
739E 1159
739E 1160                 ;*****************************************************************************
739E 1161                 ;
739E 1162                 ;
739E 1163                 ;       TEST10          OS DATA TEST
739E 1164                 ;       ------          ------------
739E 1165                 ;
739E 1166                 ;               THE 8-BIT OS REGISTER WILL BE TESTED WITH THE STANDARD DATA
739E 1167                 ;       PATTERNS OF A ONE RIPPLING THROUGH A FIELD OF ZEROS AND A ZERO RIPPLING
739E 1168                 ;       THROUGH A FIELD OF ONES.
739E 1169                 ;
739E 1170                 ;
739E 1171                 ;       TEST STEPS:
739E 1172                 ;
739E 1173                 ;               1) WRITE DATA PATTERN TO WR.
739E 1174                 ;               2) MOVE THE DATA PATTERN FROM WR TO OS.
739E 1175                 ;               3) CLEAR THE WR.
739E 1176                 ;               4) MOVE THE DATA PATTERN FROM OS TO WR.
739E 1177                 ;               5) READ THE WR.
739E 1178                 ;               6) COMPARE DATA SEND WITH DATA RETURNED.
739E 1179                 ;               7) PRINT ERRORS IF ANY.
```

```
739E 1180                                       ;               8) REPEAT 1 - 7 FOR ALL DATA PATTERNS.
739E 1181                                       ;
739E 1182                                       ;
739E 1183                                       ;       ERROR1: OS DATA TEST FAILURE.
739E 1184                                       ;
739E 1185                                       ;
739E 1186                                       ;       DEBUG:  1) CHECK THAT ''Y BUS 0 - 7 H'' IS REACHING THE OS REGISTER.
739E 1187                                       ;               2) CHECK THE DATA INTEGRETY OF THE OS REGISTER.
739E 1188                                       ;               3) CHECK THE ADDRESS DECODE FOR THE OS REGISTER ON THE.
739E 1189                                       ;                  LS ADDRESS BUS IS ''7C''.
739E 1190                                       ;               4) CHECK THE ''LOAD Y TO OS L'' SIGNAL.
739E 1191                                       ;               5) CHECK ''CM,RDEST'' FOR SIGNALS ''OS CTL 1 H'' AND
739E 1192                                       ;                  ''OS CTL 0 H''.
739E 1193                                       ;
739E 1194                                       ;
739E 1195                                       ;*****************************************************************************
739E 1196
739E 1197
739E 1198 0000'CD 10 3E                        TEST10:         HEAD    <^X10>
          0F 0000'3A
          C9 03AB'D2
          0F 0000'3A
          3D D3 0439'DA
73B4 1199
73B4 1200 008B'CD                                               CALL    LD_SUPPORT      ;LOAD SUPPORT FOR FAST READS AND
73B7 1201                                                                               ; WRITES OF WR0.
73B7 1202
73B7 1203 0055'3A                               1$:             LDA     ZERO_DAT        ;LOAD ALL ZEROS INTO DATA3 AND
73BA 1204 0000'32                                               STA     DATA3           ; INTO EXP_DAT.
73BD 1205 0072'32                                               STA     EXP_DAT
73C0 1206
73C0 1207 00'3E                                                 MVI     A,OS_ADD        ;WRITE TO THE OS REGISTER VIA WR0.
73C2 1208 00ED'CD                                               CALL    LS_WRITE
73C5 1209
73C5 1210 00'01'21                                              LXI     H,X_CLR_WR0+1   ;CLEAR WR0.
73C8 1211 01BA'CD                                               CALL    EXEC_INST
73CB 1212
73CB 1213 00'3E                                                 MVI     A,OS_ADD        ;READ THE OS REGISTER.
73CD 1214 010A'CD                                               CALL    LS_READ
73D0 1215
73D0 1216 0000'3A                                               LDA     DATA3           ;MOVE THE RECEIVED DATA TO REC_DAT.
73D3 1217 0077'32                                               STA     REC_DAT
73D6 1218
73D6 1219 00'72'21              ******************              IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'77'11        *
          03E7'CA 0000'CD       *
73E4 1220                       *                                                       ; RECEIVED DATA.
73E4 1221                       *
73E4 1222 024D'CD               *                               CALL    TEST10_ERROR    ;PRINT AN ERROR MESSAGE.
73E7 1223                       *
73E7 1224                       ******************              ENDIF
73E7 1225
73E7 1226 0000'3A                                               LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
73EA 1227 0F                                                    RRC
73EB 1228 03B7'DA                                               JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
73EE 1229
73EE 1230
73EE 1231 005A'3A                               2$:             LDA     ONE_DAT         ;LOAD ALL ONES INTO DATA3 AND
73F1 1232 0000'32                                               STA     DATA3           ; INTO EXP_DAT.
73F4 1233 0072'32                                               STA     EXP_DAT
```

```
73F7
73F7 00'3E                                              MVI     A,OS_ADD          ;WRITE TO THE OS REGISTER VIA WRO.
73F9 00ED'CD                                            CALL    LS_WRITE
73FC
73FC 00'01'21                                           LXI     H,X_CLR_WRO+1     ;CLEAR WRO.
73FF 01BA'CD                                            CALL    EXEC_INST
7402
7402 00'3E                                              MVI     A,OS_ADD          ;READ THE OS REGISTER.
7404 010A'CD                                            CALL    LS_READ
7407
7407 0000'3A                                            LDA     DATA3             ;MOVE THE RECEIVED DATA TO REC_DAT.
740A 0077'32                                            STA     REC_DAT
740D
740D 00'72'21                  ******************       IFNEQ   EXP_DAT,REC_DAT,1         ;COMPARE THE EXPECTED AND
     01 0E 00'77'11            *
     041E'CA 0000'CD           *
741B                           *                                                          ; RECEIVED DATA.
741B                           *
741B 024D'CD                   *                        CALL    TEST10_ERROR      ;PRINT AN ERROR MESSAGE.
741E                           *
741E                           ******************       ENDIF
741E
741E 0000'3A                                            LDA     ERROR_CON         ;IF THE ERROR LOOPING FLAG IS SET...
7421 0F                                                 RRC
7422 03EE'DA                                            JC      2$                ;...JUMP TO BEGINNING OF ERROR LOOP.
7425
7425
7425 0059'3A                                            LDA     ONE_SHIFT+3       ;LOAD A ONE THROUGH A ZERO FIELD
7428 006D'32                                            STA     TEMPB_DAT         ; PATTERN INTO TEMPB_DAT.
742B
742B 0257'CD                                            CALL    TEST10_SHIFT_OS ;EXECUTE TEST ON OS WITH SHIFT PATTERN
742E
742E
742E 005E'3A                                            LDA     ZERO_SHIFT+3      ;LOAD A ZERO THROUGH A ONE FIELD
7431 006D'32                                            STA     TEMPB_DAT         ; PATTERN INTO TEMPB_DAT.
7434
7434 0257'CD                                            CALL    TEST10_SHIFT_OS ;EXECUTE TEST ON OS WITH SHIFT PATTERN
7437
7437 3C D3                                              TAIL
7439
7439
7439                                           ;*************************************************************************
7439                                           ;
7439                                           ;
7439                                           ;         TEST11              LS INDEXING TEST
7439                                           ;         ------              ----------------
7439                                           ;
7439                                           ;                  TO SEE THAT THE LS CAN BE INDEXED USING THE OS, DATA
7439                                           ;         WILL BE WRITTEN INTO THE LS WITH DIRECT INSTRUCTIONS AND THEN
7439                                           ;         READ OUT USING THE INDEXED INSTRUCTION.  ONLY ADDRESSES 20 - 3F
7439                                           ;         WILL BE READ USING OS INDEXING IN THIS TEST.
7439                                           ;
7439                                           ;
7439                                           ;         TEST STEPS:
7439                                           ;
7439                                           ;                  1) WRITE BACKGROUND DATA OS SHIFTING PATTERN TO LS.
7439                                           ;                  2) WRITE OS WITH ADDRESS.
7439                                           ;                  3) READ USING OS AS INDEX.
7439                                           ;                  4) COMPARE TO SEE IF SENT DATA EQUAL RETURNED DATA.
7439                                           ;                  5) PRINT ERRORS IF ANY.
```

```
7439 1292                                        ;              6) REPEAT 1 - 5 FOR ALL OS ADDRESSES USED.
7439 1293                                        ;
7439 1294                                        ;
7439 1295                                        ;       ERROR1: LS INDEXING TEST FAILURE.
7439 1296                                        ;
7439 1297                                        ;       NOTE: ADDITIONAL DATA CONATINS PRESENT LS LOCATION.
7439 1298                                        ;
7439 1299                                        ;
7439 1300                                        ;       ASSUMPTIONS:  THE PREVIOUS TEST CHECKING THE OS HAS PASSED
7439 1301                                        ;
7439 1302                                        ;
7439 1303                                        ;       DEBUG:  1) CHECK ADDRESS DECODE FOR LS FROM THE MUX OF CSR BITS
7439 1304                                        ;                  AND OS BITS. THE OS BITS SHOULD BE ENABLED THROUGH.
7439 1305                                        ;
7439 1306                                        ;
7439 1307                                        ;*************************************************************************
7439 1308
7439 1309
7439 1310 0000'CD 11 3E                         TEST11:        HEAD    <^X11>
          0F 0000'3A
          C9 0446'D2
          0F 0000'3A
          3D D3 0500'DA
744F 1311
744F 1312 008B'CD                                               CALL    LD_SUPPORT       ;LOAD SUPPORT FOR FAST READS AND
7452 1313                                                                                ; WRITES OF WR0.
7452 1314
7452 1315 20 3E                                                 MVI     A,<^X20>         ;LOAD THE LS POINTER WITH HEX 20.
7454 1316 0007'32                                               STA     LS_ADDR
7457 1317
7457 1318 00'56'21                                              LXI     H,ONE_SHIFT      ;LOAD A 1 THROUGH A ZERO FIELD
745A 1319 00'6D'11                                              LXI     D,TEMPB_DAT      ; PATTERN INTO TEMPB_DAT
745D 1320 0000'CD                                               CALL    MOVER_4          ;MOVE THE DATA
7460 1321
7460 1322 C5 46 00'00'21    ******************                  DO      32               ;32 LS LOCATIONS (20-3F).
          77 20 3E          *
7468 1323                   *
7468 1324 00'6D'21          *                                   LXI     H,TEMPB_DAT      ;MOVE THE TEST PATTERN (SHIFTING
746B 1325 00'00'11          *                                   LXI     D,DATA0          ; ONES THROUGH ZEROS) INTO DATA0.
746E 1326 0000'CD           *                                   CALL    MOVER_4
7471 1327                   *
7471 1328 0007'3A           *                                   LDA     LS_ADDR          ;WRITE THE PATTERN TO AN LS LOCATION.
7474 1329 00ED'CD           *                                   CALL    LS_WRITE
7477 1330                   *
7477 1331 0007'3A           *                                   LDA     LS_ADDR
747A 1332 3C                *                                   INR     A                ;INCREMENT THE LS POINTER.
747B 1333 0007'32           *                                   STA     LS_ADDR
747E 1334                   *
747E 1335 B7                *                                   ORA     A                ;CLEAR CARRY
747F 1336 00'6C'21          *                                   LXI     H,TEMPB_SHIFT    ;SHIFT THE 1 THROUGH THE 0 FIELD
7482 1337 0000'22           *                                   SHLD    SHIFT_PNT        ;POINT TO TEMPB_SHIFT AREA
7485 1338 0000'CD           *                                   CALL    SHIFTER          ;SHIFT DATA
7488 1339                   *
7488 1340 35 00'00'21       ******************                  ENDDO
          70 C1 0468'C2
7491 1341
7491 1342 00'56'21                                              LXI     H,ONE_SHIFT      ;LOAD A 1 THROUGH A ZERO FIELD
7494 1343 00'6D'11                                              LXI     D,TEMPB_DAT      ; PATTERN INTO TEMPB_DAT
7497 1344 0000'CD                                               CALL    MOVER_4          ;MOVE THE DATA
749A 1345
```

```
749A 1346 0007'32 AF                                                CLEAR   LS_ADDR          ;INDEX POINTS TO LS LOCATION HEX 20.
749E 1347
749E 1348 C5 46 00'00'21         *******************                DO      32
          77 20 3E               *
74A6 1349                        *
74A6 1350 00'6D'21               *                                  LXI     H,TEMPB_DAT      ;LOAD THE TEST PATTERN INTO EXP_DAT.
74A9 1351 00'72'11               *                                  LXI     D,EXP_DAT
74AC 1352 0000'CD                *                                  CALL    MOVER_4
74AF 1353                        *
74AF 1354 0007'3A                *                        1$:       LDA     LS_ADDR          ;MOVE THE LS POINTER TO DATA3.
74B2 1355 0000'32                *                                  STA     DATA3
74B5 1356                        *
74B5 1357 00'3E                  *                                  MVI     A,OS_ADD         ;MOVE THE LS POINTER TO THE OS REG.
74B7 1358 00ED'CD                *                                  CALL    LS_WRITE
74BA 1359                        *
74BA 1360 00'22'21               *                                  LXI     H,MOV_LSOS_WR    ;WRITE FROM LS TO WR 0 USING THE OS
74BD 1361 01BA'CD                *                                  CALL    EXEC_INST        ; REG AS AN INDEX
74C0 1362                        *
74C0 1363 016E'CD                *                                  CALL    FAST_READ_WR0    ;READ WR0.
74C3 1364                        *
74C3 1365 00'00'21               *                                  LXI     H,DATA0          ;MOVE THE RECEIVED DATA TO REC_DAT.
74C6 1366 00'77'11               *                                  LXI     D,REC_DAT
74C9 1367 0000'CD                *                                  CALL    MOVER_4
74CC 1368                        *
74CC 1369 00'72'21               * ****************                 IFNEQ   EXP_DAT,REC_DAT,4        ;COMPARE THE EXPECTED AND
          04 0E 00'77'11         * *
          04DD'CA 0000'CD        * *
74DA 1370                        * *                                                                 ; RECEIVED DATA.
74DA 1371                        * *
74DA 1372 02A7'CD                * *                                CALL    TEST11_ERROR     ;PRINT AN ERROR MESSAGE.
74DD 1373                        * *
74DD 1374                        * ****************                 ENDIF
74DD 1375                        *
74DD 1376 0000'3A                *                                  LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
74E0 1377 0F                     *                                  RRC
74E1 1378 04AF'DA                *                                  JC      1$               ;...JUMP TO BEGINNING OF ERROR LOOP.
74E4 1379                        *
74E4 1380 0007'3A                *                                  LDA     LS_ADDR          ;INCREMENT THE LS POINTER.
74E7 1381 3C                     *                                  INR     A
74E8 1382 0007'32                *                                  STA     LS_ADDR
74EB 1383                        *
74EB 1384 B7                     *                                  ORA     A                ;CLEAR CARRY
74EC 1385 00'6C'21               *                                  LXI     H,TEMPB_SHIFT    ;SHIFT THE 1 THROUGH THE 0 FIELD
74EF 1386 0000'22                *                                  SHLD    SHIFT_PNT        ;POINT TO TEMPB_SHIFT AREA
74F2 1387 0000'CD                *                                  CALL    SHIFTER          ;SHIFT DATA
74F5 1388                        *
74F5 1389 35 00'00'21            *******************                ENDDO
          70 C1 04A6'C2
74FE 1390
74FE 1391 3C D3                                                     TAIL
7500 1392
7500 1393
7500 1394                                        ;*******************************************************************************
7500 1395                                        ;
7500 1396                                        ;
7500 1397                                        ;       TEST12          ALU CONDITION CODE TEST
7500 1398                                        ;       ------          -----------------------
7500 1399                                        ;
7500 1400                                        ;               THE ALU CONDITION CODES WILL BE TESTED BY LOADING THEM
7500 1401                                        ;       FROM THE LOWER FOUR BITS OF THE Y BUS. THEY WILL BE TESTED WITH
```

```
7500 1402                                       ;       THE STANDARD PATTERNS OF ZERO RIPPLED THROUGH A FIELD OF ONES AND
7500 1403                                       ;       A ONE RIPPLED THROUGH A FIELD OF ZEROS.
7500 1404                                       ;
7500 1405                                       ;
7500 1406                                       ;       TEST STEPS:
7500 1407                                       ;
7500 1408                                       ;               1) LOAD WORKING REG WITH DATA PATTERN.
7500 1409                                       ;               2) MOVE WR TO ALU CC (MOV WR TO LS[FE]).
7500 1410                                       ;               3) CLEAR WORKING REGISTER.
7500 1411                                       ;               4) MOVE ALU CC TO WR.
7500 1412                                       ;               5) READ WR TO 8085.
7500 1413                                       ;               6) COMPARE DATA SENT WITH DATA RECEIVED.
7500 1414                                       ;               7) PRINT ERROR IF ANY.
7500 1415                                       ;               8) REPEAT 1 - 7 FOR ALL PATTERNS.
7500 1416                                       ;
7500 1417                                       ;
7500 1418                                       ;       ASSUMPTIONS:  THE WORKING REGISTER WORKS AND THE CLEAR OF THAT
7500 1419                                       ;                     WORKING REGISTER WORKS.
7500 1420                                       ;
7500 1421                                       ;
7500 1422                                       ;       ERROR1: ALU CONDITION CODE TEST FAILURE.
7500 1423                                       ;
7500 1424                                       ;
7500 1425                                       ;       DEBUG:  1) CHECK ''ALU CC F0 H'' AND ''ALU CC F1 H'' FOR SELECTING
7500 1426                                       ;                  THE Y-BUS ON ''ALU N,Z'' AND ''ALU V,C''.
7500 1427                                       ;               2) CHECK THAT ''Y-BUS 03 H'' AND ''Y-BUS 02 H'' REACH ''ALU N,Z''.
7500 1428                                       ;               3) CHECK THAT ''Y-BUS 01 H'' AND ''Y-BUS 00 H'' REACH ''ALU C,V''.
7500 1429                                       ;               4) CHECK THE ''CLOCK REGS H'' SIGNAL.
7500 1430                                       ;               5) CHECK ''REG CONTROL'' TO SELECT ALU C,V,N,Z ONTO THE D-BUS.
7500 1431                                       ;
7500 1432                                       ;
7500 1433                                       ;*****************************************************************************
7500 1434
7500 1435
7500 1436
7500 1437 0000'CD 12 3E                         TEST12:         HEAD    <^X12>
          0F 0000'3A
          C9 050D'D2
          0F 0000'3A
          3D D3 05CF'DA
7516 1438
7516 1439 008B'CD                                               CALL    LD_SUPPORT      ;LOAD SUPPORT FOR FAST READS AND
7519 1440                                                                               ; WRITES OF WR0.
7519 1441
7519 1442 0059'3A                                               LDA     ONE_SHIFT+3     ;STORE THE FIRST TEST PATTERN (A ONE
751C 1443 006D'32                                               STA     TEMPB_DAT       ; THROUGH A ZERO FIELD) IN TEMPB_DAT.
751F 1444
751F 1445 C5 46 00'00'21       ******************               DO      4               ;LOW FOUR BITS ARE OF INTEREST.
          77 04 3E             *
7527 1446                      *
7527 1447 006D'3A              *                1$:             LDA     TEMPB_DAT       ;STORE PATTERN IN DATA3 AND EXP_DAT.
752A 1448 0000'32              *                                STA     DATA3
752D 1449 0072'32              *                                STA     EXP_DAT
7530 1450                      *
7530 1451 00'3E                *                                MVI     A,CC_ADDR       ;WRITE THE PATTERN TO THE CC REG.
7532 1452 00ED'CD              *                                CALL    LS_WRITE
7535 1453                      *
7535 1454 00'01'21             *                                LXI     H,X_CLR_WR0+1   ;CLEAR WR0.
7538 1455 01BA'CD              *                                CALL    EXEC_INST
753B 1456                      *
```

```
7538 1457 00'3E             *                                  MVI     A,CC_ADDR       ;READ THE PATTERN FROM THE CC REG.
753D 1458 010A'CD           *                                  CALL    LS_READ
7540 1459                   *
7540 1460 0000'3A           *                                  LDA     DATA3           ;MASK OUT THE UPPER FOUR BITS AND
7543 1461 0F E6             *                                  ANI     <^X0F>          ; MOVE THE RECEIVED DATA TO REC_DAT.
7545 1462 0077'32           *                                  STA     REC_DAT
7548 1463                   *
7548 1464 00'72'21          * *****************                IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'77'11    * *
          0559'CA 0000'CD   * *
7556 1465                   * *                                                                ; RECIEVED DATA.
7556 1466                   * *
7556 1467 02E0'CD           * *                                CALL    TEST12_ERROR    ;PRINT AN ERROR MESSAGE.
7559 1468                   * *
7559 1469                   * *****************                ENDIF
7559 1470                   *
7559 1471 0000'3A           *                                  LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
755C 1472 0F                *                                  RRC
755D 1473 0527'DA           *                                  JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7560 1474                   *
7560 1475 006D'3A           *                                  LDA     TEMPB_DAT       ;ROTATE THE ONE THROUGH THE ZERO FIELD.
7563 1476 07                *                                  RLC
7564 1477 006D'32           *                                  STA     TEMPB_DAT
7567 1478                   *
7567 1479 35 00'00'21       ******************                 ENDDO
          70 C1 0527'C2
7570 1480
7570 1481
7570 1482 005E'3A                                              LDA     ZERO_SHIFT+3    ;STORE THE SECOND TEST PATTERN (A ZERO
7573 1483 0F E6                                                ANI     <^X0F>          ; THROUGH A ONE FIELD) IN TEMPB_DAT
7575 1484 006D'32                                              STA     TEMPB_DAT       ; WITH TOP 4 BITS CLEARED.
7578 1485
7578 1486 C5 46 00'00'21    ******************                 DO      4               ;LOW FOUR BITS ARE OF INTEREST.
          77 04 3E          *
7580 1487                   *
7580 1488 006D'3A           *                        2$:       LDA     TEMPB_DAT       ;STORE PATTERN IN DATA3 AND EXP_DAT.
7583 1489 0000'32           *                                  STA     DATA3
7586 1490 0072'32           *                                  STA     EXP_DAT
7589 1491                   *
7589 1492 00'3E             *                                  MVI     A,CC_ADDR       ;WRITE THE PATTERN TO THE CC REG.
758B 1493 00ED'CD           *                                  CALL    LS_WRITE
758E 1494                   *
758E 1495 00'01'21          *                                  LXI     H,X_CLR_WR0+1   ;CLEAR WR0.
7591 1496 01BA'CD           *                                  CALL    EXEC_INST
7594 1497                   *
7594 1498 00'3E             *                                  MVI     A,CC_ADDR       ;READ THE PATTERN FROM THE CC REG.
7596 1499 010A'CD           *                                  CALL    LS_READ
7599 1500                   *
7599 1501 0000'3A           *                                  LDA     DATA3           ;MASK OUT THE UPPER FOUR BITS AND
759C 1502 0F E6             *                                  ANI     <^X0F>          ; MOVE THE RECEIVED DATA TO REC_DAT.
759E 1503 0077'32           *                                  STA     REC_DAT
75A1 1504                   *
75A1 1505 00'72'21          * *****************                IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'77'11    * *
          05B2'CA 0000'CD   * *
75AF 1506                   * *                                                                ; RECIEVED DATA.
75AF 1507                   * *
75AF 1508 02E0'CD           * *                                CALL    TEST12_ERROR    ;PRINT AN ERROR MESSAGE.
75B2 1509                   * *
75B2 1510                   * *****************                ENDIF
```

```
75B2 1511                     *
75B2 1512 0000'3A             *                               LDA      ERROR_CON      ;IF THE ERROR LOOPING FLAG IS SET...
75B5 1513 0F                  *                               RRC
75B6 1514 0580'DA             *                               JC       2$             ;...JUMP TO BEGINNING OF ERROR LOOP.
75B9 1515                     *
75B9 1516 006D'3A             *                               LDA      TEMPB_DAT      ;ROTATE THE ZERO THROUGH THE ONE FIELD.
75BC 1517 07                  *                               RLC                     ;ONLY THE FOUR LOW BITS ARE OF INTEREST
75BD 1518 0F E6               *                               ANI      <^X0F>         ; SO CLEAR THE HIGH FOUR BITS AND SET
75BF 1519 01 F6               *                               ORI      <^X01>         ; THE LSB.
75C1 1520 006D'32             *                               STA      TEMPB_DAT
75C4 1521                     *
75C4 1522 35 00'00'21         ******************              ENDDO
          70 C1 0580'C2
75CD 1523
75CD 1524
75CD 1525 3C D3                                               TAIL
75CF 1526
75CF 1527
75CF 1528                                     ;****************************************************************************
75CF 1529                                     ;
75CF 1530                                     ;       TEST13          INC WORKING REGISTER TEST
75CF 1531                                     ;       ------          -------------------------
75CF 1532                                     ;
75CF 1533                                     ;               THE INCREMENT INSTRUCTION WILL BE TESTED BY CLEARING
75CF 1534                                     ;       A WR AND EXECUTING THE INC INSTRUCTION AND READING THE REGISTER.
75CF 1535                                     ;
75CF 1536                                     ;       TEST STEPS:
75CF 1537                                     ;
75CF 1538                                     ;
75CF 1539                                     ;               1) CLEAR WORKING REGISTER.
75CF 1540                                     ;               2) EXECUTE INC INSTRUCTION.
75CF 1541                                     ;               3) READ A 1 FROM THE WR.
75CF 1542                                     ;               4) PRINT ERRORS IF ANY.
75CF 1543                                     ;
75CF 1544                                     ;       ASSUMPTIONS:  THE WORKING REGISTERS ARE WORKING AS TESTED
75CF 1545                                     ;                     IN PREVIOUS TESTS.
75CF 1546                                     ;
75CF 1547                                     ;
75CF 1548                                     ;       ERROR1: INC WORKING REGISTER TEST FAILURE.
75CF 1549                                     ;
75CF 1550                                     ;
75CF 1551                                     ;       DEBUG:  1) CHECK ''SRC CTL H'' FOR ''O.A'' = 4.
75CF 1552                                     ;               2) CHECK ''ALU CTL H'' FOR ''R.PLUS.S'' = 0.
75CF 1553                                     ;               3) CHECK ''DEST CTL H'' FOR ''WRITE B.F'' = 3.
75CF 1554                                     ;               4) CHECK ''ALU CARRY-IN H'' FOR HIGH VALUE.
75CF 1555                                     ;
75CF 1556                                     ;
75CF 1557                                     ;****************************************************************************
75CF 1558
75CF 1559
75CF 1560 0000'CD 13 3E                       TEST13:         HEAD     <^X13>
          0F 0000'3A
          C9 05DC'D2
          0F 0000'3A
          3D D3 0620'DA
75E5 1561
75E5 1562 00'56'21                                            LXI      H,ONE_SHIFT    ;LOAD A ONE INTO EXP_DAT, WHICH
75E8 1563 00'72'11                                            LXI      D,EXP_DAT      ; SHOULD RESULT FROM INCREMENTING
75EB 1564 0000'CD                                             CALL     MOVER_4        ; THE CLEARED WR[0].
75EE 1565
```

```
75EE 1566 00'01'21                   1$:                 LXI     H,X_CLR_WRO+1   ;CLEAR WRO.
75F1 1567 01BA'CD                                        CALL    EXEC_INST
75F4 1568
75F4 1569 00'25'21                                       LXI     H,INC_WRO       ;INCREMENT WRO.
75F7 1570 01BA'CD                                        CALL    EXEC_INST
75FA 1571
75FA 1572 016E'CD                                        CALL    FAST_READ_WRO   ;READ WRO.
75FD 1573
75FD 1574 00'00'21                                       LXI     H,DATA0         ;MOVE THE RECEIVED DATA TO REC_DAT.
7600 1575 00'77'11                                       LXI     D,REC_DAT
7603 1576 0000'CD                                        CALL    MOVER_4
7606 1577
7606 1578 00'72'21           *******************         IFNEQ   EXP_DAT,REC_DAT,4       ;COMPARE THE EXPECTED AND
          04 0E 00'77'11     *
          0617'CA 0000'CD    *
7614 1579                    *                                                           ; RECEIVED DATA.
7614 1580                    *
7614 1581 02EA'CD            *                           CALL    TEST13_ERROR    ;PRINT AN ERROR MESSAGE.
7617 1582                    *
7617 1583                    *******************         ENDIF
7617 1584
7617 1585 0000'3A                                       LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
761A 1586 0F                                             RRC
761B 1587 05EE'DA                                        JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
761E 1588
761E 1589 3C D3                                          TAIL
7620 1590
7620 1591
7620 1592                           ;*******************************************************************************
7620 1593                           :
7620 1594                           :
7620 1595                           :        TEST14          Z BIT LONG WORD TEST
7620 1596                           :        ------          --------------------
7620 1597                           :
7620 1598                           :                A 2901 REAGISTER WILL BE CLEARED WITH LOAD CC AND THEN THE
7620 1599                           :        ALU CC WILL BE READ AND THE Z BIT WILL BE VERIFIED TO BE ON. THE
7620 1600                           :        REGISTER WILL NEXT BE INCREMENTED WITH LOAD CC AND THEN THE
7620 1601                           :        ALU CC WILL BE READ AND THE Z BIT WILL BE VERIFIED TO ON OFF.
7620 1602                           :
7620 1603                           :
7620 1604                           :        TEST STEPS:
7620 1605                           :
7620 1606                           :                1) CLEAR WR, LOAD ALU CC.
7620 1607                           :                2) READ ALU CC AND VERIFY Z BIT ON.
7620 1608                           :                3) PRINT ERROR IF ANY.
7620 1609                           :                4) INC WR, LOAD ALU CC.
7620 1610                           :                5) READ ALU CC AND VERIFY Z BIT OFF.
7620 1611                           :                6) PRINT ERROR IF ANY.
7620 1612                           :
7620 1613                           :
7620 1614                           :        ASSUMPTIONS:  IT IS ASSUMED THAT THE ALU CONDITION CODE BITS HAVE
7620 1615                           :                      BEEN TESTED AND ARE WORKING FROM THE LAST TEST AND THAT
7620 1616                           :                      THERE IS NO PROBLEM IN READING THEM (ALSO FROM THE LAST
7620 1617                           :                      TEST).
7620 1618                           :
7620 1619                           :
7620 1620                           :        ERROR1: Z BIT LONG WORD TEST FAILURE, Z BIT NOT SET.
7620 1621                           :        ERROR2: Z BIT LONG WORD TEST FAILURE, Z BIT NOT CLEAR.
7620 1622                           :
7620 1623                           :
```

```
7620 1624                                             ;          DEBUG:  1) CHECK ALU N,Z PAL FOR ''Z LONG H'' FROM THE 2901 TO
7620 1625                                             ;                     ALU N,Z.
7620 1626                                             ;                  2) CHECK ''ALU CC F0 H'' AND ''ALU CC F1 H'' FOR SELECTING
7620 1627                                             ;                     Z LONG H'' ON ''ALU N,Z''.
7620 1628                                             ;
7620 1629                                             ;
7620 1630                                             ;*******************************************************************************
7620 1631
7620 1632
7620 1633 0000'CD 14 3E                               TEST14:         HEAD    <^X14>
          0F 0000'3A
          C9 062D'D2
          0F 0000'3A
          3D D3 06A8'DA
7636 1634
7636 1635 01 3E                                                       MVI     A,1                 ;FIRST PART OF TEST 14.
7638 1636 0000'32                                                     STA     CON_ERR_NUM
763B 1637
763B 1638 04 3E                                                       MVI     A,<^X04>            ;EXPECTED DATA HAS ZERO BIT SET.
763D 1639 0072'32                                                     STA     EXP_DAT
7640 1640
7640 1641 00'2B'21                                    1$:             LXI     H,CLR_WR0_CC        ;CLEAR WR0 AND SET FLAGS APPROPRIATELY.
7643 1642 01BA'CD                                                     CALL    EXEC_INST
7646 1643
7646 1644 00'3E                                                       MVI     A,CC_ADDR           ;READ CC REG.
7648 1645 010A'CD                                                     CALL    LS_READ
764B 1646
764B 1647 0000'3A                                                     LDA     DATA3               ;MASK OUT ALL BITS BUT THE CARRY BIT
764E 1648 04 E6                                                       ANI     <^X04>              ; AND MOVE RECEIVED DATA TO REC_DAT.
7650 1649 0077'32                                                     STA     REC_DAT
7653 1650
7653 1651 00'72'21                 ******************                 IFNEQ   EXP_DAT,REC_DAT,1          ;COMPARE THE EXPECTED AND
          01 0E 00'77'11           *
          0664'CA 0000'CD          *
7661 1652                          *                                                                     ; RECEIVED DATA.
7661 1653                          *
7661 1654 0311'CD                  *                                  CALL    TEST14_15_ERROR ;PRINT AN ERROR MESSAGE.
7664 1655                          *
7664 1656                          ******************                 ENDIF
7664 1657
7664 1658 0000'3A                                                     LDA     ERROR_CON           ;IF THE ERROR LOOPING FLAG IS SET...
7667 1659 0F                                                          RRC
7668 1660 0640'DA                                                     JC      1$                  ;...JUMP TO BEGINNING OF ERROR LOOP.
766B 1661
766B 1662
766B 1663 02 3E                                                       MVI     A,2                 ;SECOND PART OF TEST 14.
766D 1664 0000'32                                                     STA     CON_ERR_NUM
7670 1665
7670 1666 00 3E                                                       MVI     A,0                 ;EXPECTED DATA HAS ZERO BIT CLEARED.
7672 1667 0072'32                                                     STA     EXP_DAT
7675 1668
7675 1669 00'2B'21                                    2$:             LXI     H,CLR_WR0_CC        ;CLEAR WR0 AND SET FLAGS APPROPRIATELY.
7678 1670 01BA'CD                                                     CALL    EXEC_INST
767B 1671
767B 1672 00'28'21                                                    LXI     H,INC_WR0_CC        ;INCREMENT WR0 AND SET FLAGS
767E 1673 01BA'CD                                                     CALL    EXEC_INST           ; APPROPRIATELY.
7681 1674
7681 1675 00'3E                                                       MVI     A,CC_ADDR           ;READ CC REG.
7683 1676 010A'CD                                                     CALL    LS_READ
7686 1677
```

```
7686 1678 0000'3A                                    LDA     DATA3           ;MASK OUT ALL BITS BUT THE CARRY BIT
7689 1679 04 E6                                      ANI     <^X04>          ; AND MOVE RECEIVED DATA TO REC_DAT.
768B 1680 0077'32                                    STA     REC_DAT
768E 1681
768E 1682 00'72'21          ******************       IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'77'11    *
          069F'CA 0000'CD   *
769C 1683                   *                                                        ; RECEIVED DATA.
769C 1684                   *
769C 1685 0311'CD           *                        CALL    TEST14_15_ERROR ;PRINT AN ERROR MESSAGE.
769F 1686                   *
769F 1687                   ******************       ENDIF
769F 1688
769F 1689 0000'3A                                    LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
76A2 1690 0F                                         RRC
76A3 1691 0675'DA                                    JC      2$              ;...JUMP TO BEGINNING OF ERROR LOOP.
76A6 1692
76A6 1693 3C D3                                      TAIL
76A8 1694
76A8 1695
76A8 1696                                 ;*******************************************************************************
76A8 1697                                 ;
76A8 1698                                 ;
76A8 1699                                 ;       TEST15          2901 XOR (WR WITH WR) Z BIT SET/CLEAR TEST
76A8 1700                                 ;       ------          ------------------------------------------
76A8 1701                                 ;
76A8 1702                                 ;
76A8 1703                                 ;               THE 2901 XOR FUNCTION WILL BE TESTED USING THE MOVE
76A8 1704                                 ;       INSTRUCTION (XOR WR[] WITH WR[] TO Q).  THE TEST WILL CONSIST
76A8 1705                                 ;       OF XORING THE FOLLOWING TABLE FOR EACH BIT POSITION.  AFTER
76A8 1706                                 ;       EACH XOR THE Z BIT WILL BE CHECKED FOR THE PROPER VALUE.
76A8 1707                                 ;
76A8 1708                                 ;               WR0     WR1
76A8 1709                                 ;               0       0
76A8 1710                                 ;               0       1
76A8 1711                                 ;               1       0
76A8 1712                                 ;               1       1
76A8 1713                                 ;
76A8 1714                                 ;       TEST STEPS:
76A8 1715                                 ;
76A8 1716                                 ;               1) LOAD WORKING REGISTERS.
76A8 1717                                 ;               2) PERFORM XOR FUNCTION,LOAD ALU CC.
76A8 1718                                 ;               3) READ ZBIT.
76A8 1719                                 ;               4) COMPARE WITH EXPECTED VALUE.
76A8 1720                                 ;               5) PRINT ERRORS IF ANY.
76A8 1721                                 ;               6) REPEAT 1 - 5 FOR ALL PATTERNS.
76A8 1722                                 ;
76A8 1723                                 ;
76A8 1724                                 ;       ASSUMPTIONS:  THE WORKING REGISTERS AND THE DATA PATH TO THE
76A8 1725                                 ;                     2901 ARE WORKING AS TESTED IN PREVIOUS TESTS.
76A8 1726                                 ;
76A8 1727                                 ;
76A8 1728                                 ;       ERROR1: XOR Z BIT SET/CLEAR TEST FAILURE (WR0=WR1=0).
76A8 1729                                 ;       ERROR2: XOR Z BIT SET/CLEAR TEST FAILURE (WR0=0,WR1=1).
76A8 1730                                 ;       ERROR3: XOR Z BIT SET/CLEAR TEST FAILURE (WR0=1,WR1=0).
76A8 1731                                 ;       ERROR4: XOR Z BIT SET/CLEAR TEST FAILURE (WR0=WR1=1).
76A8 1732                                 ;
76A8 1733                                 ;
76A8 1734                                 ;       DEBUG:  1) CHECK 2901 FOR PROPER XOR FUNCTION ''ALU CTL H'' = 6
76A8 1735                                 ;               2) CHECK ''Z-LONG H''
```

```
76A8 1736                                                ;
76A8 1737                                                ;*****************************************************************************
76A8 1738
76A8 1739
76A8 1740
76A8 1741 0000'CD 15 3E                                   TEST15:         HEAD    <^X15>
          0F 0000'3A
          C9 06B5'D2
          0F 0000'3A
          3D D3 07DC'DA
76BE 1742
76BE 1743 01 3E                                                          MVI     A,1             ;FIRST PART OF TEST 15.
76C0 1744 0000'32                                                        STA     CON_ERR_NUM
76C3 1745
76C3 1746 04 3E                                                          MVI     A,<^X04>        ;ZERO BIT SET IN EXP_DAT.
76C5 1747 0072'32                                                        STA     EXP_DAT
76C8 1748
76C8 1749 00'01'21                                       1$:             LXI     H,X_CLR_WR0+1   ;CLEAR WR0.
76CB 1750 01BA'CD                                                        CALL    EXEC_INST
76CE 1751
76CE 1752 00'2E'21                                                       LXI     H,X_CLR_WR1     ;CLEAR WR1.
76D1 1753 01BA'CD                                                        CALL    EXEC_INST
76D4 1754
76D4 1755 00'34'21                                                       LXI     H,XOR_WR0_WR1   ;XOR WR0 WITH WR1 AND SET FLAGS.
76D7 1756 01BA'CD                                                        CALL    EXEC_INST
76DA 1757
76DA 1758 00'3E                                                          MVI     A,CC_ADDR       ;READ CC REG.
76DC 1759 010A'CD                                                        CALL    LS_READ
76DF 1760
76DF 1761 0000'3A                                                        LDA     DATA3           ;MASK OUT ALL BITS BUT THE CARRY BIT
76E2 1762 04 E6                                                          ANI     <^X04>          ; AND MOVE RECEIVED DATA TO REC_DAT.
76E4 1763 0077'32                                                        STA     REC_DAT
76E7 1764
76E7 1765 00'72'21                   ******************                  IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'77'11             *
          06F8'CA 0000'CD            *
76F5 1766                            *                                                                   ; RECEIVED DATA.
76F5 1767                            *
76F5 1768 0311'CD                    *                                   CALL    TEST14_15_ERROR ;PRINT AN ERROR MESSAGE.
76F8 1769                            *
76F8 1770                            ******************                  ENDIF
76F8 1771
76F8 1772 0000'3A                                                        LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
76FB 1773 0F                                                             RRC
76FC 1774 06C8'DA                                                        JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
76FF 1775
76FF 1776
76FF 1777 02 3E                                                          MVI     A,2             ;SECOND PART OF TEST 15.
7701 1778 0000'32                                                        STA     CON_ERR_NUM
7704 1779
7704 1780 00 3E                                                          MVI     A,0             ;ZERO BIT CLEARED IN EXP_DAT.
7706 1781 0072'32                                                        STA     EXP_DAT
7709 1782
7709 1783 00'01'21                                       2$:             LXI     H,X_CLR_WR0+1   ;CLEAR WR0.
770C 1784 01BA'CD                                                        CALL    EXEC_INST
770F 1785
770F 1786 00'2E'21                                                       LXI     H,X_CLR_WR1     ;CLEAR WR1.
7712 1787 01BA'CD                                                        CALL    EXEC_INST
7715 1788
7715 1789 00'31'21                                                       LXI     H,X_COM_WR1     ;COMPLEMENT WR1.
```

```
7718 1790 01BA'CD                                          CALL    EXEC_INST
771B 1791
771B 1792 00'34'21                                         LXI     H,XOR_WRO_WR1   ;XOR WRO WITH WR1 AND SET FLAGS.
771E 1793 01BA'CD                                          CALL    EXEC_INST
7721 1794
7721 1795 00'3E                                            MVI     A,CC_ADDR       ;READ CC REG.
7723 1796 010A'CD                                          CALL    LS_READ
7726 1797
7726 1798 0000'3A                                          LDA     DATA3           ;MASK OUT ALL BITS BUT THE CARRY BIT
7729 1799 04 E6                                            ANI     <^X04>          ; AND MOVE RECEIVED DATA TO REC_DAT.
772B 1800 0077'32                                          STA     REC_DAT
772E 1801
772E 1802 00'72'21          ******************             IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'77'11    *
          073F'CA 0000'CD   *
773C 1803                   *                                                              ; RECEIVED DATA.
773C 1804                   *
773C 1805 0311'CD           *                              CALL    TEST14_15_ERROR ;PRINT AN ERROR MESSAGE.
773F 1806                   *
773F 1807                   ******************             ENDIF
773F 1808
773F 1809 0000'3A                                          LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7742 1810 0F                                               RRC
7743 1811 0709'DA                                          JC      2$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7746 1812
7746 1813
7746 1814
7746 1815 03 3E                                            MVI     A,3             ;THIRD PART OF TEST 15.
7748 1816 0000'32                                          STA     CON_ERR_NUM
774B 1817
774B 1818 00 3E                                            MVI     A,0             ;ZERO BIT CLEARED IN EXP_DAT.
774D 1819 0072'32                                          STA     EXP_DAT
7750 1820
7750 1821 00'01'21                                 3$:     LXI     H,X_CLR_WRO+1   ;CLEAR WRO.
7753 1822 01BA'CD                                          CALL    EXEC_INST
7756 1823
7756 1824 00'00'21                                         LXI     H,X_COM_WRO     ;COMPLEMENT WRO.
7759 1825 01BA'CD                                          CALL    EXEC_INST
775C 1826
775C 1827 00'2E'21                                         LXI     H,X_CLR_WR1     ;CLEAR WR1.
775F 1828 01BA'CD                                          CALL    EXEC_INST
7762 1829
7762 1830 00'34'21                                         LXI     H,XOR_WRO_WR1   ;XOR WRO WITH WR1 AND SET FLAGS.
7765 1831 01BA'CD                                          CALL    EXEC_INST
7768 1832
7768 1833 00'3E                                            MVI     A,CC_ADDR       ;READ CC REG.
776A 1834 010A'CD                                          CALL    LS_READ
776D 1835
776D 1836 0000'3A                                          LDA     DATA3           ;MASK OUT ALL BITS BUT THE CARRY BIT
7770 1837 04 E6                                            ANI     <^X04>          ; AND MOVE RECEIVED DATA TO REC_DAT.
7772 1838 0077'32                                          STA     REC_DAT
7775 1839
7775 1840 00'72'21          ******************             IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'77'11    *
          0786'CA 0000'CD   *
7783 1841                   *                                                              ; RECEIVED DATA.
7783 1842                   *
7783 1843 0311'CD           *                              CALL    TEST14_15_ERROR ;PRINT AN ERROR MESSAGE.
7786 1844                   *
7786 1845                   ******************             ENDIF
```

```
7786 1846
7786 1847 0000'3A                                             LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
7789 1848 0F                                                  RRC
778A 1849 0750'DA                                             JC      3$               ;...JUMP TO BEGINNING OF ERROR LOOP.
778D 1850
778D 1851
778D 1852
778D 1853 04 3E                                               MVI     A,4              ;FOURTH PART OF TEST 15.
778F 1854 0000'32                                             STA     CON_ERR_NUM
7792 1855
7792 1856 04 3E                                               MVI     A,<^X04>         ;ZERO BIT SET IN EXP_DAT.
7794 1857 0072'32                                             STA     EXP_DAT
7797 1858
7797 1859 00'01'21                         4$:                LXI     H,X_CLR_WR0+1    ;CLEAR WR0.
779A 1860 01BA'CD                                             CALL    EXEC_INST
779D 1861
779D 1862 00'00'21                                            LXI     H,X_COM_WR0      ;COMPLEMENT WR0.
77A0 1863 01BA'CD                                             CALL    EXEC_INST
77A3 1864
77A3 1865 00'2E'21                                            LXI     H,X_CLR_WR1      ;CLEAR WR1.
77A6 1866 01BA'CD                                             CALL    EXEC_INST
77A9 1867
77A9 1868 00'31'21                                            LXI     H,X_COM_WR1      ;COMPLEMENT WR1.
77AC 1869 01BA'CD                                             CALL    EXEC_INST
77AF 1870
77AF 1871 00'34'21                                            LXI     H,XOR_WR0_WR1    ;XOR WR0 WITH WR1 AND SET FLAGS.
77B2 1872 01BA'CD                                             CALL    EXEC_INST
77B5 1873
77B5 1874 00'3E                                               MVI     A,CC_ADDR        ;READ CC REG.
77B7 1875 010A'CD                                             CALL    LS_READ
77BA 1876
77BA 1877 0000'3A                                             LDA     DATA3            ;MASK OUT ALL BITS BUT THE CARRY BIT
77BD 1878 04 E6                                               ANI     <^X04>           ; AND MOVE RECEIVED DATA TO REC_DAT.
77BF 1879 0077'32                                             STA     REC_DAT
77C2 1880
77C2 1881 00'72'21                 *******************        IFNEQ   EXP_DAT,REC_DAT,1        ;COMPARE THE EXPECTED AND
          01 0E 00'77'11           *
          07D3'CA 0000'CD          *
77D0 1882                          *                                                           ; RECEIVED DATA.
77D0 1883                          *
77D0 1884 0311'CD                  *                          CALL    TEST14_15_ERROR ;PRINT AN ERROR MESSAGE.
77D3 1885                          *
77D3 1886                          *******************        ENDIF
77D3 1887
77D3 1888 0000'3A                                             LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
77D6 1889 0F                                                  RRC
77D7 1890 0797'DA                                             JC      4$               ;...JUMP TO BEGINNING OF ERROR LOOP.
77DA 1891
77DA 1892 3C D3                                               TAIL
77DC 1893
77DC 1894
77DC 1895                          ;*****************************************************************************
77DC 1896                          :
77DC 1897                          :
77DC 1898                          :      TEST16          2901 XOR (WR WITH LS) Z BIT SET/CLEAR TEST
77DC 1899                          :      ------          ------------------------------------------
77DC 1900                          :
77DC 1901                          :              THE 2901 XOR FUNCTION WILL BE TESTED USING THE MOVE
77DC 1902                          :      INSTRUCTION (XOR WR[] WITH LS[] TO Q).  THE TEST WILL CONSIST
77DC 1903                          :      OF XORING THE FOLLOWING TABLE FOR EACH BIT POSITION.  AFTER
```

```
77DC 1904                                  ;       EACH XOR THE Z BIT WILL BE CHECKED FOR THE PROPER VALUE.
77DC 1905                                  ;
77DC 1906                                  ;               WR      LS
77DC 1907                                  ;               0       0
77DC 1908                                  ;               0       1
77DC 1909                                  ;               1       0
77DC 1910                                  ;               1       1
77DC 1911                                  ;
77DC 1912                                  ;
77DC 1913                                  ;       TEST STEPS:
77DC 1914                                  ;
77DC 1915                                  ;               1) LOAD WORKING REGISTER AND LS LOCATION.
77DC 1916                                  ;               2) PERFORM XOR FUNCTION, LOAD ALU CC.
77DC 1917                                  ;               3) READ Z BIT.
77DC 1918                                  ;               4) COMPARE WITH EXPECTED VALUE.
77DC 1919                                  ;               5) PRINT ERRORS IF ANY.
77DC 1920                                  ;               6) REPEAT 1 - 5 FOR ALL PATTERNS.
77DC 1921                                  ;
77DC 1922                                  ;
77DC 1923                                  ;       ASSUMPTIONS:  THE WORKING REGISTERS AND THE DATA PATH TO THE
77DC 1924                                  ;                     2901 ARE WORKING AS TESTED IN PREVIOUS TESTS.
77DC 1925                                  ;
77DC 1926                                  ;
77DC 1927                                  ;       ERROR1: XOR Z BIT SET/CLEAR TEST FAILURE (LS=WR=0).
77DC 1928                                  ;       ERROR2: XOR Z BIT SET/CLEAR TEST FAILURE (LS=0,WR=1).
77DC 1929                                  ;       ERROR3: XOR Z BIT SET/CLEAR TEST FAILURE (LS=1,WR=0).
77DC 1930                                  ;       ERROR4: XOR Z BIT SET/CLEAR TEST FAILURE (LS=WR=1).
77DC 1931                                  ;
77DC 1932                                  ;
77DC 1933                                  ;       DEBUG:  1) CHECK 2901 FOR PROPER XOR FUNCTION ''ALU CTL H'' = 6.
77DC 1934                                  ;               2) CHECK ''Z-LONG H''.
77DC 1935                                  ;
77DC 1936                                  ;
77DC 1937                                  ;***************************************************************************
77DC 1938
77DC 1939
77DC 1940
77DC 1941 0000'CD 16 3E                    TEST16:         HEAD    <^X16>
          0F 0000'3A
          C9 07E9'D2
          0F 0000'3A
          3D D3 0927'DA
77F2 1942
77F2 1943 008B'CD                                          CALL    LD_SUPPORT       ;LOAD SUPPORT FOR FAST READS AND
77F5 1944                                                                           ; WRITES OF WR0.
77F5 1945
77F5 1946 01 3E                                            MVI     A,1              ;FIRST PART OF TEST 16.
77F7 1947 0000'32                                          STA     CON_ERR_NUM
77FA 1948
77FA 1949 04 3E                                            MVI     A,<^X04>         ;ZERO BIT SET IN EXP_DAT.
77FC 1950 0072'32                                          STA     EXP_DAT
77FF 1951
77FF 1952 00'55'21                         1$:             LXI     H,ZERO_DAT       ;MOVE ALL ZEROS TO DATA0.
7802 1953 00'00'11                                         LXI     D,DATA0
7805 1954 0000'CD                                          CALL    MOVER_4
7808 1955
7808 1956 00 3E                                            MVI     A,0              ;WRITE DATA TO LS0.
780A 1957 00ED'CD                                          CALL    LS_WRITE
780D 1958
780D 1959 00'01'21                                         LXI     H,X_CLR_WR0+1    ;CLEAR WR0.
```

```
7810 1960 01BA'CD                                       CALL    EXEC_INST
7813 1961
7813 1962 00'37'21                                      LXI     H,XOR_WRO_LSO   ;XOR THE CONTENTS OF WRO AND LSO
7816 1963 01BA'CD                                       CALL    EXEC_INST       ; AND SET THE CC FLAGS.
7819 1964
7819 1965 00'3E                                         MVI     A,CC_ADDR       ;READ CC REG.
781B 1966 010A'CD                                       CALL    LS_READ
781E 1967
781E 1968 0000'3A                                       LDA     DATA3           ;MASK OUT ALL BITS BUT THE CARRY BIT
7821 1969 04 E6                                         ANI     <^X04>          ; AND MOVE RECEIVED DATA TO REC_DAT.
7823 1970 0077'32                                       STA     REC_DAT
7826 1971
7826 1972 00'72'21          ******************          IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'77'11    *
          0837'CA 0000'CD   *
7834 1973                   *                                                           ; RECEIVED DATA.
7834 1974                   *
7834 1975 031B'CD           *                           CALL    TEST16_ERROR    ;PRINT AN ERROR MESSAGE.
7837 1976                   *
7837 1977                   ******************          ENDIF
7837 1978
7837 1979 0000'3A                                       LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
783A 1980 0F                                            RRC
783B 1981 07FF'DA                                       JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
783E 1982
783E 1983
783E 1984
783E 1985 02 3E                                         MVI     A,2             ;SECOND PART OF TEST 16.
7840 1986 0000'32                                       STA     CON_ERR_NUM
7843 1987
7843 1988 00 3E                                         MVI     A,0             ;ZERO BIT CLEARED IN EXP_DAT.
7845 1989 0072'32                                       STA     EXP_DAT
7848 1990
7848 1991 00'55'21                              2$:     LXI     H,ZERO_DAT      ;MOVE ALL ZERO TO DATA0.
784B 1992 00'00'11                                      LXI     D,DATA0
784E 1993 0000'CD                                       CALL    MOVER_4
7851 1994
7851 1995 00 3E                                         MVI     A,0             ;WRITE DATA TO LS0.
7853 1996 00ED'CD                                       CALL    LS_WRITE
7856 1997
7856 1998 00'01'21                                      LXI     H,X_CLR_WRO+1   ;CLEAR WRO.
7859 1999 01BA'CD                                       CALL    EXEC_INST
785C 2000
785C 2001 00'00'21                                      LXI     H,X_COM_WRO     ;COMPLEMENT WRO.
785F 2002 01BA'CD                                       CALL    EXEC_INST
7862 2003
7862 2004 00'37'21                                      LXI     H,XOR_WRO_LSO   ;XOR THE CONTENTS OF WRO AND LSO
7865 2005 01BA'CD                                       CALL    EXEC_INST       ; AND SET THE CC FLAGS.
7868 2006
7868 2007 00'3E                                         MVI     A,CC_ADDR       ;READ CC REG.
786A 2008 010A'CD                                       CALL    LS_READ
786D 2009
786D 2010 0000'3A                                       LDA     DATA3           ;MASK OUT ALL BITS BUT THE CARRY BIT
7870 2011 04 E6                                         ANI     <^X04>          ; AND MOVE RECEIVED DATA TO REC_DAT.
7872 2012 0077'32                                       STA     REC_DAT
7875 2013
7875 2014 00'72'21          ******************          IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'77'11    *
          0886'CA 0000'CD   *
7883 2015                   *                                                           ; RECEIVED DATA.
```

```
7883 2016                        *
7883 2017 031B'CD                *                     CALL    TEST16_ERROR    ;PRINT AN ERROR MESSAGE.
7886 2018                        *
7886 2019                        ******************    ENDIF
7886 2020
7886 2021 0000'3A                                      LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7889 2022 0F                                           RRC
788A 2023 0848'DA                                      JC      2$              ;...JUMP TO BEGINNING OF ERROR LOOP.
788D 2024
788D 2025
788D 2026
788D 2027 03 3E                                        MVI     A,3             ;THIRD PART OF TEST 16.
788F 2028 0000'32                                      STA     CON_ERR_NUM
7892 2029
7892 2030 00 3E                                        MVI     A,0             ;ZERO BIT CLEARED IN EXP_DAT.
7894 2031 0072'32                                      STA     EXP_DAT
7897 2032
7897 2033 00'5A'21                          3$:        LXI     H,ONE_DAT       ;MOVE ALL ONES TO DATA0.
789A 2034 00'00'11                                     LXI     D,DATA0
789D 2035 0000'CD                                      CALL    MOVER_4
78A0 2036
78A0 2037 00 3E                                        MVI     A,0             ;WRITE DATA TO LS0.
78A2 2038 00ED'CD                                      CALL    LS_WRITE
78A5 2039
78A5 2040 00'01'21                                     LXI     H,X_CLR_WR0+1   ;CLEAR WR0.
78A8 2041 01BA'CD                                      CALL    EXEC_INST
78AB 2042
78AB 2043 00'37'21                                     LXI     H,XOR_WR0_LS0   ;XOR THE CONTENTS OF WR0 AND LS0
78AE 2044 01BA'CD                                      CALL    EXEC_INST       ; AND SET THE CC FLAGS.
78B1 2045
78B1 2046 00'3E                                        MVI     A,CC_ADDR       ;READ CC REG.
78B3 2047 010A'CD                                      CALL    LS_READ
78B6 2048
78B6 2049 0000'3A                                      LDA     DATA3           ;MASK OUT ALL BITS BUT THE CARRY BIT
78B9 2050 04 E6                                        ANI     <^X04>          ; AND MOVE RECEIVED DATA TO REC_DAT.
78BB 2051 0077'32                                      STA     REC_DAT
78BE 2052
78BE 2053 00'72'21               ******************    IFNEQ   EXP_DAT,REC_DAT,1        ;COMPARE THE EXPECTED AND
          01 0E 00'77'11         *
          08CF'CA 0000'CD        *
78CC 2054                        *                                                      ; RECEIVED DATA.
78CC 2055                        *
78CC 2056 031B'CD                *                     CALL    TEST16_ERROR    ;PRINT AN ERROR MESSAGE.
78CF 2057                        *
78CF 2058                        ******************    ENDIF
78CF 2059
78CF 2060 0000'3A                                      LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
78D2 2061 0F                                           RRC
78D3 2062 0897'DA                                      JC      3$              ;...JUMP TO BEGINNING OF ERROR LOOP.
78D6 2063
78D6 2064
78D6 2065
78D6 2066 04 3E                                        MVI     A,4             ;FOURTH PART OF TEST 16.
78D8 2067 0000'32                                      STA     CON_ERR_NUM
78DB 2068
78DB 2069 04 3E                                        MVI     A,<^X04>        ;ZERO BIT SET IN EXP_DAT.
78DD 2070 0072'32                                      STA     EXP_DAT
78E0 2071
78E0 2072 00'5A'21                          4$:        LXI     H,ONE_DAT       ;MOVE ALL ONES TO DATA0.
78E3 2073 00'00'11                                     LXI     D,DATA0
```

```
78E6 2074 0000'CD                                        CALL    MOVER_4
78E9 2075
78E9 2076 00 3E                                          MVI     A,0             ;WRITE DATA TO LSO.
78EB 2077 00ED'CD                                        CALL    LS_WRITE
78EE 2078
78EE 2079 00'01'21                                       LXI     H,X_CLR_WRO+1   ;CLEAR WRO.
78F1 2080 01BA'CD                                        CALL    EXEC_INST
78F4 2081
78F4 2082 00'00'21                                       LXI     H,X_COM_WRO     ;COMPLEMENT WRO.
78F7 2083 01BA'CD                                        CALL    EXEC_INST
78FA 2084
78FA 2085 00'37'21                                       LXI     H,XOR_WRO_LSO   ;XOR THE CONTENTS OF WRO AND LSO
78FD 2086 01BA'CD                                        CALL    EXEC_INST       ; AND SET THE CC FLAGS.
7900 2087
7900 2088 00'3E                                          MVI     A,CC_ADDR       ;READ CC REG.
7902 2089 010A'CD                                        CALL    LS_READ
7905 2090
7905 2091 0000'3A                                        LDA     DATA3           ;MASK OUT ALL BITS BUT THE CARRY BIT
7908 2092 04 E6                                          ANI     <^X04>          ; AND MOVE RECEIVED DATA TO REC_DAT.
790A 2093 0077'32                                        STA     REC_DAT
790D 2094
790D 2095 00'72'21           ******************          IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'77'11     *
          091E'CA 0000'CD    *
791B 2096                    *                                                           ; RECEIVED DATA.
791B 2097                    *
791B 2098 031B'CD            *                           CALL    TEST16_ERROR    ;PRINT AN ERROR MESSAGE.
791E 2099                    *
791E 2100                    ******************          ENDIF
791E 2101
791E 2102 0000'3A                                        LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7921 2103 0F                                             RRC
7922 2104 08E0'DA                                        JC      4$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7925 2105
7925 2106 3C D3                                          TAIL
7927 2107
7927 2108
7927 2109                                      ;**************************************************************************
7927 2110                                      ;
7927 2111                                      ;
7927 2112                                      ;       TEST17          CPU ACK SET/CLEAR
7927 2113                                      ;       ------          -----------------
7927 2114                                      ;
7927 2115                                      ;               CPU ACK WILL BE CLEARED SO THAT IT WILL BE IN KNOWN
7927 2116                                      ;       STATE BUT NOT CHECKED. THEN IT WILL BE SET SO THAT THE SET
7927 2117                                      ;       CAN BE CHECKED.  ONCE THE SET IS CHECKED IT WILL BE CLEARED
7927 2118                                      ;       TO TEST THE CLEAR.
7927 2119                                      ;
7927 2120                                      ;
7927 2121                                      ;       TEST STEPS:
7927 2122                                      ;
7927 2123                                      ;               1) LOAD CSR WITH MISC INST TO CLEAR CPU ACK AND EXECUTE.
7927 2124                                      ;               2) LOAD CSR WITH MISC INST TO SET CPU ACK AND EXECUTE.
7927 2125                                      ;               3) READ CPU ACK TO VERIFY IT IS SET.
7927 2126                                      ;               4) PRINT ERROR IF ANY.
7927 2127                                      ;               5) LOAD CSR WITH MISC INST TO CLEAR CPU ACK AND EXECUTE.
7927 2128                                      ;               6) READ CPU ACK TO VERIFY IT IS CLEAR.
7927 2129                                      ;               7) PRINT ERROR IF ANY.
7927 2130                                      ;
7927 2131                                      ;
```

```
7927 2132                                      ;       ERROR1: CPU ACK SET FAILURE.
7927 2133                                      ;       ERROR2: CPU ACK CLEAR FAILURE.
7927 2134                                      ;
7927 2135                                      ;
7927 2136                                      ;       DEBUG:  1) CHECK THAT 'MISC INSTR H'' IS PRESENT ON THE
7927 2137                                      ;                  STATE REG. MISC DECODER.
7927 2138                                      ;               2) CHECK CSR BITS 12 - 15,18 ARE REACHING THE
7927 2139                                      ;                  STATE REG. MISC DECODER.
7927 2140                                      ;               3) CHECK ''IRD STATE L''.
7927 2141                                      ;               4) CHECK STATE REG. MISC DECODER PAL TO SEE IF SET OR CLEARED.
7927 2142                                      ;
7927 2143                                      ;
7927 2144                                      ;*****************************************************************************
7927 2145
7927 2146
7927 2147
7927 2148 0000'CD 17 3E                        TEST17:         HEAD    <^X17>
          0F 0000'3A
          C9 0934'D2
          0F 0000'3A
          3D D3 09A2'DA
793D 2149
793D 2150 01 3E                                                MVI     A,1             ;FIRST PART OF TEST 17.
793F 2151 0000'32                                              STA     CON_ERR_NUM
7942 2152
7942 2153 0072'32 3C AF                                        SET     EXP_DAT         ;LOAD SET CPU ACK INTO EXP_DAT.
7947 2154
7947 2155 00'3A'21                             1$:             LXI     H,CLR_CPU_ACK   ;CLEAR CPU ACK.
794A 2156 01BA'CD                                              CALL    EXEC_INST
794D 2157
794D 2158 00'3D'21                                             LXI     H,SET_CPU_ACK   ;SET CPU ACK.
7950 2159 01BA'CD                                              CALL    EXEC_INST
7953 2160
7953 2161 82 DB                                                IN      CPUACK          ;READ CPU ACK.
7955 2162
7955 2163 01 E6                                                ANI     <^X01>          ;MASK OUT OTHER BITS.
7957 2164
7957 2165 0077'32                                              STA     REC_DAT         ;STORE RECEIVED DATA IN REC_DAT.
795A 2166
795A 2167 00'72'21          *******************                IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE EXPECTED AND
          01 0E 00'77'11    *
          096B'CA 0000'CD   *
7968 2168                   *                                                                   ; RECEIVED DATA.
7968 2169                   *
7968 2170 0325'CD           *                                  CALL    TEST17_18_ERROR ;PRINT AN ERROR MESSAGE.
796B 2171                   *
796B 2172                   *******************                ENDIF
796B 2173
796B 2174 0000'3A                                              LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
796E 2175 0F                                                   RRC
796F 2176 0947'DA                                              JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7972 2177
7972 2178
7972 2179 02 3E                                                MVI     A,2             ;SECOND PART OF TEST 17.
7974 2180 0000'32                                              STA     CON_ERR_NUM
7977 2181
7977 2182 0072'32 AF                                           CLEAR   EXP_DAT         ;LOAD CLEAR CPU ACK INTO EXP_DAT.
797B 2183
797B 2184 00'3A'21                             2$:             LXI     H,CLR_CPU_ACK   ;CLEAR CPU ACK.
797E 2185 01BA'CD                                              CALL    EXEC_INST
```

```
7981
7981 82 DB                                     IN      CPUACK          ;READ CPU ACK.
7983
7983 01 E6                                     ANI     <^X01>          ;MASK OUT OTHER BITS.
7985
7985 0077'32                                   STA     REC_DAT         ;STORE RECEIVED DATA IN REC_DAT.
7988
7988 00'72'21           ******************     IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE EXPECTED AND
     01 0E 00'77'11     *
     0999'CA 0000'CD    *
7996                    *                                                       ; RECEIVED DATA.
7996                    *
7996 0325'CD            *                      CALL    TEST17_18_ERROR ;PRINT AN ERROR MESSAGE.
7999                    *
7999                    ******************     ENDIF
7999
7999 0000'3A                                   LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
799C 0F                                        RRC
799D 097B'DA                                   JC      2$              ;...JUMP TO BEGINNING OF ERROR LOOP.
79A0
79A0 3C D3                                     TAIL
79A2
79A2
79A2                            ;*******************************************************************************
79A2                            ;
79A2                            ;
79A2                            ;       TEST18          CPU ATTN SET/CLEAR
79A2                            ;       ------          ------------------
79A2                            ;
79A2                            ;               CPU ATTN WILL BE CLEARED SO THAT IT WILL BE IN KNOWN
79A2                            ;       STATE BUT NOT CHECKED. THEN IT WILL BE SET SO THAT THE SET
79A2                            ;       CAN BE CHECKED.  ONCE THE SET IS CHECKED IT WILL BE CLEARED
79A2                            ;       TO TEST THE CLEAR.
79A2                            ;
79A2                            ;
79A2                            ;       TEST STEPS:
79A2                            ;
79A2                            ;               1) LOAD CSR WITH MISC INST TO CLEAR CPU ATTN AND EXECUTE.
79A2                            ;               2) LOAD CSR WITH MISC INST TO SET CPU ATTN AND EXECUTE.
79A2                            ;               3) READ CPU ATTN TO VERIFY IT IS SET.
79A2                            ;               4) PRINT ERROR IF ANY.
79A2                            ;               5) LOAD CSR WITH MISC INST TO CLEAR CPU ATTN AND EXECUTE.
79A2                            ;               6) READ CPU ATTN TO VERIFY IT IS CLEAR.
79A2                            ;               7) PRINT ERROR IF ANY.
79A2                            ;
79A2                            ;
79A2                            ;       ERROR1: CPU ATTN SET FAILURE.
79A2                            ;       ERROR2: CPU ATTN CLEAR FAILURE.
79A2                            ;
79A2                            ;
79A2                            ;       DEBUG:  1) CHECK THAT ''MISC INSTR H'' IS PRESENT ON THE
79A2                            ;                  STATE REG. MISC DECODER.
79A2                            ;               2) CHECK CSR BITS 12 - 15,18 ARE REACHING THE
79A2                            ;                  STATE REG. MISC DECODER.
79A2                            ;               3) CHECK ''IRD STATE L''.
79A2                            ;               4) CHECK STATE REG. MISC DECODER PAL TO SEE IF SET OR CLEARED.
79A2                            ;
79A2                            ;
79A2                            ;*******************************************************************************
79A2
```

```
79A2 2244
79A2 2245
79A2 2246
79A2 2247 0000'CD 18 3E                   TEST18:         HEAD    <^X18>
          0F 0000'3A
          C9 09AF'D2
          0F 0000'3A
          3D D3 0A1D'DA
79B8 2248
79B8 2249 01 3E                                           MVI     A,1             ;FIRST PART OF TEST 18.
79BA 2250 0000'32                                         STA     CON_ERR_NUM
79BD 2251
79BD 2252 0072'32 3C AF                                   SET     EXP_DAT         ;LOAD SET CPU ATTN INTO EXP_DAT.
79C2 2253
79C2 2254 00'40'21                        1$:             LXI     H,CLR_CPU_ATTN  ;CLEAR CPU ATTN.
79C5 2255 01BA'CD                                         CALL    EXEC_INST
79C8 2256
79C8 2257 00'43'21                                        LXI     H,SET_CPU_ATTN  ;SET CPU ATTN.
79CB 2258 01BA'CD                                         CALL    EXEC_INST
79CE 2259
79CE 2260 83 DB                                           IN      CPATTN          ;READ CPU ATTN.
79D0 2261
79D0 2262 01 E6                                           ANI     <^X01>          ;MASK OUT OTHER BITS.
79D2 2263
79D2 2264 0077'32                                         STA     REC_DAT         ;STORE RECEIVED DATA IN REC_DAT.
79D5 2265
79D5 2266 00'72'21          ******************            IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE EXPECTED AND
          01 0E 00'77'11    *
          09E6'CA 0000'CD   *
79E3 2267                   *                                                             ; RECEIVED DATA.
79E3 2268                   *
79E3 2269 0325'CD           *                             CALL    TEST17_18_ERROR ;PRINT AN ERROR MESSAGE.
79E6 2270                   *
79E6 2271                   ******************            ENDIF
79E6 2272
79E6 2273 0000'3A                                         LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
79E9 2274 0F                                              RRC
79EA 2275 09C2'DA                                         JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
79ED 2276
79ED 2277 02 3E                                           MVI     A,2             ;SECOND PART OF TEST 18.
79EF 2278 0000'32                                         STA     CON_ERR_NUM
79F2 2279
79F2 2280 0072'32 AF                                      CLEAR   EXP_DAT         ;LOAD CLEAR CPU ATTN INTO EXP_DAT.
79F6 2281
79F6 2282 00'40'21                        2$:             LXI     H,CLR_CPU_ATTN  ;CLEAR CPU ATTN.
79F9 2283 01BA'CD                                         CALL    EXEC_INST
79FC 2284
79FC 2285 83 DB                                           IN      CPATTN          ;READ CPU ATTN.
79FE 2286
79FE 2287 01 E6                                           ANI     <^X01>          ;MASK OUT OTHER BITS.
7A00 2288
7A00 2289 0077'32                                         STA     REC_DAT         ;STORE RECEIVED DATA IN REC_DAT.
7A03 2290
7A03 2291 00'72'21          ******************            IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE EXPECTED AND
          01 0E 00'77'11    *
          0A14'CA 0000'CD   *
7A11 2292                   *                                                             ; RECEIVED DATA.
7A11 2293                   *
7A11 2294 0325'CD           *                             CALL    TEST17_18_ERROR ;PRINT AN ERROR MESSAGE.
7A14 2295                   *
```

```
7A14 2296                   ******************          ENDIF
7A14 2297
7A14 2298 0000'3A                                       LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7A17 2299 0F                                            RRC
7A18 2300 09F6'DA                                       JC      2$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7A1B 2301
7A1B 2302 3C D3                                         TAIL
7A1D 2303
7A1D 2304
7A1D 2305
7A1D 2306                              ;*******************************************************************************
7A1D 2307                              ;
7A1D 2308                              ;
7A1D 2309                              ;       TEST19          CLOCK STALL
7A1D 2310                              ;       ------          -----------
7A1D 2311                              ;
7A1D 2312                              ;               THE WCS WILL BE LOADED WITH THE FOLLOWING INSTRUCTIONS:
7A1D 2313                              ;
7A1D 2314                              ;               NOP
7A1D 2315                              ;               NOP
7A1D 2316                              ;               NOP     (WITH PARITY ERROR)
7A1D 2317                              ;               NOP
7A1D 2318                              ;               JMP  *
7A1D 2319                              ;
7A1D 2320                              ;       ENABLE PARITY ERROR WILL BE TURNED ON AND THE CPU STARTED.  THE CPU
7A1D 2321                              ;       SHOULD STOP AT THE ''NOP'' WITH THE PARTIY ERROR AND CAUSE A STALL AND
7A1D 2322                              ;       INTERRUPT.  AFTER ALLOWING THE CPU TO RUN LONG ENOUGH TO DETECT THE
7A1D 2323                              ;       PARITY ERROR, THE UPC WILL BE READ.  IF ALL WENT WELL, THE UPC SHOULD
7A1D 2324                              ;       POINT TO THE INSTRUCTION AFTER THE NOP WITH PARITY ERROR (WCS ADDRESS
7A1D 2325                              ;       3).  IF NO STALL AND INTERRUPT OCCURS, THE UPC WILL POINT TO THE
7A1D 2326                              ;       JMP * INSTRUCTION.
7A1D 2327                              ;
7A1D 2328                              ;
7A1D 2329                              ;
7A1D 2330                              ;       TEST STEPS:
7A1D 2331                              ;
7A1D 2332                              ;               1) LOAD NOP'S AND PARITY ERROR.
7A1D 2333                              ;               2) START CPU.
7A1D 2334                              ;               3) WAIT FOR INTERRUPT OR TIMEOUT.
7A1D 2335                              ;               4) READ UPC.
7A1D 2336                              ;               5) PRINT ERRORS IF ANY.
7A1D 2337                              ;
7A1D 2338                              ;       ASSUMPTIONS:  THE PARITY CHECKING LOGIC WORKS AS TESTED BY
7A1D 2339                              ;                     PREVIOUS TESTS.
7A1D 2340                              ;
7A1D 2341                              ;
7A1D 2342                              ;       ERROR1: CPU CLOCK STALL ON PARITY ERROR FAILURE.
7A1D 2343                              ;
7A1D 2344                              ;
7A1D 2345                              ;       DEBUG:  1) CHECK ''HALT ON PE L'' AND ''CS PARITY ERR H''.
7A1D 2346                              ;               2) CHECK THE GATES ORING ''CS PARITY ERR H'' TO ''CLOCK STALL L''.
7A1D 2347                              ;
7A1D 2348                              ;
7A1D 2349                              ;*******************************************************************************
7A1D 2350
7A1D 2351
7A1D 2352 19 3E                        TEST19:         MVI     A,<^X19>        ;TEST 19 INDICATOR.
7A1F 2353
7A1F 2354 0000'CD                                       CALL    CON_BEGIN_TEST  ;REPORT BEGINING OF TEST
7A22 2355
```

```
7A22 2356 0F 0000'3A         ******************                IF      EOS_FLG          ;END OF SECTION
          0A2A'D2            *
7A29 2357                    *
7A29 2358 C9                 *                                 RET
7A2A 2359                    *
7A2A 2360                    ******************                ENDIF
7A2A 2361
7A2A 2362 0F 0000'3A         ******************                IF      SKIP_TEST        ;IS TEST TO BE SKIPPED.
          0A34'D2            *
7A31 2363                    *
7A31 2364 0ADD'C3            *                                 JMP     END_TEST19       ;GO TO END OF TEST
7A34 2365                    *
7A34 2366                    ******************                ENDIF
7A34 2367
7A34 2368 3D D3                                                OUT     SETLIT           ;TURN ON THE RUN LIGHT.
7A36 2369
7A36 2370 0A3C'C3                                              JMP     TEST19_PAR_ERR+3   ;SKIP NEXT INSTRUCTION.
7A39 2371
7A39 2372 0AAA'C3                              TEST19_PAR_ERR: JMP     2$               ;THIS INSTRUCTION IS USED IN
7A3C 2373                                                                               ; RETURNING FROM A PARITY ERROR.
7A3C 2374
7A3C 2375 03 3E                                                MVI     A,3              ;UPC SHOULD POINT TO WCS[3] AFTER
7A3E 2376 0072'32                                              STA     EXP_DAT          ; PARITY INTERRUPT.
7A41 2377
7A41 2378 00'00'21                                             LXI     H,NOP_INST       ;LOAD NOP INSTRUCTION INTO WCS_VALUE.
7A44 2379 00'00'11                                             LXI     D,WCS_VALUE
7A47 2380 0000'CD                                              CALL    MOVER_3
7A4A 2381
7A4A 2382 0000'32 3C AF                                        SET     WCS_BAD_PARITY   ;LOAD NOP WITH BAD PARITY INTO WCS[2].
7A4F 2383 00 3E                                                MVI     A,0
7A51 2384 0000'32                                              STA     WCS_ADDRESS
7A54 2385 02 3E                                                MVI     A,2
7A56 2386 0001'32                                              STA     WCS_ADDRESS+1
7A59 2387 0000'CD                                              CALL    WRITE_WCS
7A5C 2388
7A5C 2389 007B'CD                                              CALL    CHECK_RD         ;CALL BOOT CODE IF UNDER RD CONTROL
7A5F 2390
7A5F 2391 00 3E                                                MVI     A,0              ;LOAD NOPS WITH GOOD PARITY INTO WCS
7A61 2392 0001'32                                              STA     WCS_ADDRESS+1    ; LOCATIONS 0,1 AND 3.
7A64 2393 0000'CD                                              CALL    WRITE_WCS
7A67 2394
7A67 2395 007B'CD                                              CALL    CHECK_RD         ;CALL BOOT CODE IF UNDER RD CONTROL
7A6A 2396
7A6A 2397 01 3E                                                MVI     A,1
7A6C 2398 0001'32                                              STA     WCS_ADDRESS+1
7A6F 2399 0000'CD                                              CALL    WRITE_WCS
7A72 2400
7A72 2401 007B'CD                                              CALL    CHECK_RD         ;CALL BOOT CODE IF UNDER RD CONTROL
7A75 2402
7A75 2403 03 3E                                                MVI     A,3
7A77 2404 0001'32                                              STA     WCS_ADDRESS+1
7A7A 2405 0000'CD                                              CALL    WRITE_WCS
7A7D 2406
7A7D 2407 007B'CD                                              CALL    CHECK_RD         ;CALL BOOT CODE IF UNDER RD CONTROL
7A80 2408
7A80 2409 00'49'21                                             LXI     H,JUMP_WCS4      ;LOAD JUMP WCS[4] INTO WCS[4].
7A83 2410 00'00'11                                             LXI     D,WCS_VALUE
7A86 2411 0000'CD                                              CALL    MOVER_3
7A89 2412
7A89 2413 04 3E                                                MVI     A,4
```

```
7A8B 2414 0001'32                                               STA    WCS_ADDRESS+1
7A8E 2415 0000'CD                                               CALL   WRITE_WCS
7A91 2416
7A91 2417 00'5F'21                                              LXI    H,TEST19_DAT     ;PREPARE TO RETURN FROM A PARITY ERROR.
7A94 2418 4C 22 11                                              LXI    D,PARITY_VECTOR
7A97 2419 0000'CD                                               CALL   MOVER_3
7A9A 2420
7A9A 2421 007B'CD                               1$:             CALL   CHECK_RD         ;CALL BOOT CODE IF UNDER RD CONTROL
7A9D 2422
7A9D 2423 A0 D3                                                 OUT    ENBPAR           ;ENABLE PARITY ERROR.
7A9F 2424
7A9F 2425 00 00 21                                              LXI    H,0              ;START EXECUTION AT LOCATION ZERO.
7AA2 2426 0000'22                                               SHLD   UPC_VALUE
7AA5 2427 0000'CD                                               CALL   LOAD_UPC
7AA8 2428
7AA8 2429 21 D3                                                 OUT    SETCLK           ;START THE CPU.
7AAA 2430
7AAA 2431 00                                    2$:             NOP                     ;ALLOW TIME FOR PARITY ERROR TO OCCUR.
7AAB 2432
7AAB 2433 20 D3                                                 OUT    CLRCLK           ;STOP THE CLOCK (IF IT IS NOT ALREADY
7AAD 2434                                                                               ; STOPPED).
7AAD 2435
7AAD 2436 A1 D3                                                 OUT    DISPAR           ;DISABLE HALT ON PARITY ERROR.
7AAF 2437
7AAF 2438 FB                                                    EI                      ;ENABLE FURTHER INTERRUPTS.
7AB0 2439
7AB0 2440 C1                                                    POP    B                ;RESTORE THE STACK.
7AB1 2441
7AB1 2442 0000'CD                                               CALL   LOAD_UPC         ;LOAD THE UPC INTO UPC_VALUE.
7AB4 2443
7AB4 2444 0001'3A                                               LDA    UPC_VALUE+1      ;MOVE THE UPC VALUE TO REC_DAT.
7AB7 2445 0077'32                                               STA    REC_DAT
7ABA 2446
7ABA 2447 00'72'21           ******************                 IFNEQ  EXP_DAT,REC_DAT,1        ;COMPARE THE EXPECTED AND
          01 0E 00'77'11     *
          0ACB'CA 0000'CD    *
7AC8 2448                    *                                                                  ; RECEIVED DATA.
7AC8 2449                    *
7AC8 2450 032F'CD            *                                  CALL   TEST19_ERROR     ;PRINT AN ERROR MESSAGE.
7ACB 2451                    *
7ACB 2452                    ******************                 ENDIF
7ACB 2453
7ACB 2454 0000'3A                                               LDA    ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
7ACE 2455 0F                                                    RRC
7ACF 2456 0A9A'DA                                               JC     1$               ;...JUMP TO BEGINNING OF ERROR LOOP.
7AD2 2457
7AD2 2458 3C D3                                                 OUT    CLRLIT           ;TURN OF THE RUN LIGHT.
7AD4 2459
7AD4 2460 00'00'21                                              LXI    H,PARITY_JUMP    ;RESTORE THE PARITY VECTOR TO ITS
7AD7 2461 4C 22 11                                              LXI    D,PARITY_VECTOR ; VALUE.
7ADA 2462 0000'CD                                               CALL   MOVER_3
7ADD 2463
7ADD 2464 00                                    END_TEST19:     NOP
7ADE 2465
7ADE 2466
7ADE 2467                                       ;*****************************************************************************
7ADE 2468                                       ;
7ADE 2469                                       ;
7ADE 2470                                       ;       TEST1A         DC LOW
7ADE 2471                                       ;       ------         -- ---
```

```
7ADE 2472                                      ;
7ADE 2473                                      ;               THE SIGNAL ''UNIBUS DC LOW L'' WILL BE ASSERTED FROM THE
7ADE 2474                                      ;       CONSOLE AND THEN READ BACK FROM THE CONSOLE AS ''SEE DC LO H''.
7ADE 2475                                      ;
7ADE 2476                                      ;       TEST STEPS:
7ADE 2477                                      ;
7ADE 2478                                      ;               1) ASSERT ''UNIBUS DC LO L''.
7ADE 2479                                      ;               2) READ ''SEE DC LO H''.
7ADE 2480                                      ;               3) CLR ''UNIBUS DC LO L''.
7ADE 2481                                      ;               4) PRINT ERROR IF ANY.
7ADE 2482                                      ;
7ADE 2483                                      ;       ERROR1:  UNIBUS DC LO FAILURE.
7ADE 2484                                      ;
7ADE 2485                                      ;       DEBUG:  1) CHECK THE LATCH HOLDING ''UNIBUS DC LO L''.
7ADE 2486                                      ;               2) CHECK A9,A10,A11 FOR 0.
7ADE 2487                                      ;               3) CHECK ''WRT M1 L''.
7ADE 2488                                      ;
7ADE 2489                                      ;
7ADE 2490                                      ;*******************************************************************************
7ADE 2491
7ADE 2492
7ADE 2493 0000'CD 1A 3E                        TEST1A:         HEAD    <^X1A>
          0F 0000'3A
          C9 0AEB'D2
          0F 0000'3A
          3D D3 0B21'DA
7AF4 2494
7AF4 2495 80 3E                                                MVI     A,<^X80>        ;LOAD BYTE WITH MSB SET INTO
7AF6 2496 0072'32                                              STA     EXP_DAT         ; EXP_DAT.
7AF9 2497
7AF9 2498 30 D3                                1$:             OUT     SETDCL          ;ASSERT DC LOW SIGNAL.
7AFB 2499
7AFB 2500 02 DB                                                IN      DCLO            ;READ DC LOW SIGNAL.
7AFD 2501 80 E6                                                ANI     <^X80>          ;MASK OUT ALL BITS BUT THE MSB
7AFF 2502 0077'32                                              STA     REC_DAT         ; AND STORE IN REC_DAT.
7B02 2503
7B02 2504 31 D3                                                OUT     CLRDCL          ;CLEAR THE DC LOW SIGNAL.
7B04 2505
7B04 2506 00'72'21          ******************                 IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'77'11    *
          0B15'CA 0000'CD   *
7B12 2507                   *                                                                  ; RECEIVED DATA.
7B12 2508                   *
7B12 2509 0339'CD           *                                  CALL    TEST1A_ERROR    ;PRINT AN ERROR MESSAGE.
7B15 2510                   *
7B15 2511                   ******************                 ENDIF
7B15 2512
7B15 2513 007B'CD                                              CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7B18 2514
7B18 2515 0000'3A                                              LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7B1B 2516 0F                                                   RRC
7B1C 2517 0AF9'DA                                              JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7B1F 2518
7B1F 2519 3C D3                                                TAIL
7B21 2520
7B21 2521
7B21 2522                                      ;*******************************************************************************
7B21 2523                                      ;
7B21 2524                                      ;
7B21 2525                                      ;       TEST1B          AC LOW
```

```
7B21 2526                                       ;       ------          ------
7B21 2527                                       ;
7B21 2528                                       ;               THE SIGNAL ''UNIBUS AC LOW L'' WILL BE ASSERTED FROM
7B21 2529                                       ;       THE CONSOLE AND THEN GO INTO A TIMING LOOP WAITING FOR AN 8085
7B21 2530                                       ;       INTERRUPT.
7B21 2531                                       ;
7B21 2532                                       ;       TEST STEPS:
7B21 2533                                       ;
7B21 2534                                       ;               1) ASSERT ''UNIBUS AC LO L''.
7B21 2535                                       ;               2) LOOP UNTIL TIMOUT OR INTERRUPT.
7B21 2536                                       ;               3) CLR ''UNIBUS AC LO L''.
7B21 2537                                       ;               4) PRINT ERROR IF NO INTERRUPT.
7B21 2538                                       ;               5) IF THERE IS AN INTERRUPT THEN CHECK THAT THE 8085 SID PIN
7B21 2539                                       ;                  IS SET AND CLEARED ALONG WITH ''UNIBUS AC LO L''. THE
7B21 2540                                       ;                  SID PIN IS USED BY THE CONSOLE TO EXIT FROM THE CONSOLE
7B21 2541                                       ;                  POWER FAIL CODE.
7B21 2542                                       ;
7B21 2543                                       ;
7B21 2544                                       ;       ERROR1:  UNIBUS AC LO FAILURE.
7B21 2545                                       ;       ERROR2:  SID WAS NOT SET BY AC LO.
7B21 2546                                       ;       ERROR3:  SID WAS NOT CLEARED BY AC LO.
7B21 2547                                       ;
7B21 2548                                       ;       DEBUG:
7B21 2549                                       ;
7B21 2550                                       ;          FOR ''UNIBUS AC LO L'' SIGNAL
7B21 2551                                       ;
7B21 2552                                       ;               1) CHECK THE LATCH HOLDING ''UNIBUS AC LO L''.
7B21 2553                                       ;               2) CHECK A9,A10,A11 FOR 1.
7B21 2554                                       ;               3) CHECK ''WRT M1 L''.
7B21 2555                                       ;               4) CHECK THE 8085 RST 7.5 PIN FOR A RISING EDGE WHEN
7B21 2556                                       ;                  ''UNIBUS AC LO L'' IS ASSERTED.
7B21 2557                                       ;
7B21 2558                                       ;          FOR 8085 SID SIGNAL
7B21 2559                                       ;
7B21 2560                                       ;               1) CHECK THE 8085 SID SIGNAL FOR A HIGH AFTER ''UNIBUS AC LO L''
7B21 2561                                       ;                  IS ASSERTED AND A LOW AFTER ''UNIBUS AC LO L'' IS NEGATED.
7B21 2562                                       ;
7B21 2563                                       ;
7B21 2564                                       ;**************************************************************************
7B21 2565
7B21 2566
7B21 2567
7B21 2568 1B 3E                                 TEST1B:         MVI     A,<^X1B>        ;TEST 1B INDICATOR.
7B23 2569
7B23 2570 0000'CD                                               CALL    CON_BEGIN_TEST  ;REPORT BEGINING OF TEST
7B26 2571
7B26 2572 0F 0000'3A    ******************                      IF      EOS_FLG         ;END OF SECTION
          0B2E'D2       *
7B2D 2573               *
7B2D 2574 C9            *                                       RET
7B2E 2575               *
7B2E 2576               ******************                      ENDIF
7B2E 2577
7B2E 2578 0F 0000'3A    ******************                      IF      SKIP_TEST       ;IS TEST TO BE SKIPPED.
          0B38'D2       *
7B35 2579               *
7B35 2580 0BEE'C3       *                                       JMP     END_TEST1B      ;GO TO END OF TEST
7B38 2581               *
7B38 2582               ******************                      ENDIF
7B38 2583
```

```
7B38 2584 3D D3                                          OUT     SETLIT            ;TURN ON THE RUN LIGHT.
7B3A 2585
7B3A 2586 00'46'21                       ERROR_LOOP:     LXI     H,JUMP_WCS0       ;LOAD A JUMP WCS[0] INTO WCS[0].
7B3D 2587 00'00'11                                       LXI     D,WCS_VALUE
7B40 2588 0000'CD                                        CALL    MOVER_3
7B43 2589
7B43 2590 00 00 21                                       LXI     H,0
7B46 2591 0000'22                                        SHLD    WCS_ADDRESS
7B49 2592 0000'CD                                        CALL    WRITE_WCS
7B4C 2593
7B4C 2594 00 00 21                                       LXI     H,0               ;LOAD THE UPC WITH 0.
7B4F 2595 0000'22                                        SHLD    UPC_VALUE
7B52 2596 0000'CD                                        CALL    LOAD_UPC
7B55 2597
7B55 2598 00'62'21                                       LXI     H,TEST1B_DAT      ;PREPARE TO RETURN FROM A TRAP.
7B58 2599 4C 25 11                                       LXI     D,POWER_VECTOR
7B5B 2600 0000'CD                                        CALL    MOVER_3
7B5E 2601
7B5E 2602 007B'CD                                        CALL    CHECK_RD          ;CALL BOOT CODE IF UNDER RD CONTROL
7B61 2603
7B61 2604 21 D3                                          OUT     SETCLK            ;STAPT THE CPU.
7B63 2605
7B63 2606 32 D3                                          OUT     SETACL            ;ASSERT AC LOW L.
7B65 2607
7B65 2608 00                                             NOP                       ;ALLOW TIME FOR A TRAP TO OCCUR.
7B66 2609
7B66 2610 20 D3                                          OUT     CLRCLK            ;STOP THE CPU.
7B68 2611
7B68 2612 33 D3                                          OUT     CLRACL            ;CLEAR AC LOW.
7B6A 2613
7B6A 2614 0343'CD                                        CALL    TEST1B_ERROR      ;PRINT AN ERROR.
7B6D 2615
7B6D 2616 0000'3A                                        LDA     ERROR_CON         ;IF THE ERROR LOOPING FLAG IS SET...
7B70 2617 0F                                             RRC
7B71 2618 0B3A'DA                                        JC      ERROR_LOOP        ;...JUMP TO BEGINNING OF ERROR LOOP.
7B74 2619
7B74 2620 0BEE'C3                                        JMP     END_TEST1B        ;JUMP TO END OF TEST.
7B77 2621
7377 2622 20 D3                          TEST1B_TRAP:    OUT     CLRCLK            ;STOP THE CPU.
7B79 2623
7B79 2624 02 3E                                          MVI     A,2               ;SECOND PART OF TEST 1B.
7B7B 2625 0000'32                                        STA     CON_ERR_NUM
7B7E 2626
7B7E 2627 80 3E                                          MVI     A,<^X80>          ;LOAD THE 8085 SID SET INTO EXP_DAT.
7B80 2628 0072'32                                        STA     EXP_DAT
7B83 2629
7B83 2630 20                                             RIM                       ;READ THE 8085 INTERRUPT STATUS AND
7B84 2631 80 E6                                          ANI     <^X80>            ; MASK OUT SID.
7B86 2632
7B86 2633 0077'32                                        STA     REC_DAT           ;STORE SID VALUE IN REC_DAT.
7B89 2634
7B89 2635 33 D3                                          OUT     CLRACL            ;CLEAR AC LOW.
7B8B 2636
7B8B 2637 00'72'21            ********************       IFNEQ   EXP_DAT,REC_DAT,1         ;COMPARE THE EXPECTED AND
          01 0E 00'77'11      *
          0B9C'CA 0000'CD     *
7B99 2638                     *
7B99 2639                     *
7B99 2640                     *                                                            ;RECEIVED DATA.
7B99 2641                     *
```

```
7B99 2642 0352'CD           *                             CALL    TEST1B_2_3_ERROR        ;PRINT AN ERROR MESSAGE.
7B9C 2643                   *
7B9C 2644                   ******************            ENDIF
7B9C 2645
7B9C 2646 0000'3A                                         LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7B9F 2647 0F                                              RRC
7BA0 2648 0B3A'DA                                         JC      ERROR_LOOP      ;...JUMP TO BEGINNING OF ERROR LOOP.
7BA3 2649
7BA3 2650 C5 46 00'00'21    ******************            DO      80              ;CALL CHECK_RD 80 TIMES IN LOOP
          77 50 3E          *
7BAB 2651                   *
7BAB 2652 03 00 21          *                             LXI     H,<^X300>       ;ENTER DELAY LOOP TO ALLOW AC
7BAE 2653 2B                *               1$:           DCX     H               ; LOW TO SETTLE.
7BAF 2654 7C                *                             MOV     A,H
7BB0 2655 B5                *                             ORA     L
7BB1 2656 0BAE'C2           *                             JNZ     1$
7BB4 2657                   *
7BB4 2658 007B'CD           *                             CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7BB7 2659                   *
7BB7 2660 35 00'00'21       ******************            ENDDO
          70 C1 0BAB'C2
7BC0 2661
7BC0 2662 03 3E                                           MVI     A,3             ;THIRD PART OF TEST 1B.
7BC2 2663 0000'32                                         STA     CON_ERR_NUM
7BC5 2664
7BC5 2665 00 3E                                           MVI     A,<^X00>        ;LOAD THE 8085 SID CLEARED
7BC7 2666 0072'32                                         STA     EXP_DAT         ; INTO EXP_DAT.
7BCA 2667
7BCA 2668 20                                              RIM                     ;READ THE 8085 INTERRUPT STATUS AND
7BCB 2669 80 E6                                           ANI     <^X80>          ; MASK OUT SID.
7BCD 2670
7BCD 2671 0077'32                                         STA     REC_DAT         ;STORE SID VALUE IN REC_DAT.
7BD0 2672
7BD0 2673 00'72'21          ******************            IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'77'11    *
          0BE1'CA 0000'CD   *
7BDE 2674                   *
7BDE 2675                   *
7BDE 2676                   *                                                     ;RECEIVED DATA.
7BDE 2677                   *
7BDE 2678 0352'CD           *                             CALL    TEST1B_2_3_ERROR        ;PRINT AN ERROR MESSAGE.
7BE1 2679                   *
7BE1 2680                   ******************            ENDIF
7BE1 2681
7BE1 2682 0000'3A                                         LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7BE4 2683 0F                                              RRC
7BE5 2684 0B3A'DA                                         JC      ERROR_LOOP      ;...JUMP TO BEGINNING OF ERROR LOOP.
7BE8 2685
7BE8 2686 10 3E                                           MVI     A,HEX_10        ;MOVE A HEX 10 TO A.
7BEA 2687 30                                              SIM                     ;RESET 7.5.
7BEB 2688
7BEB 2689 C1                                              POP     B               ;RESTORE THE STACK, WHICH WAS CHANGED
7BEC 2690 C1                                              POP     B               ; DURING THE INTERRUPT.
7BED 2691
7BED 2692 FB                                              EI                      ;ENABLE FURTHER INTERRUPTS.
7BEE 2693
7BEE 2694 3C D3                             END_TEST1B:   OUT     CLRLIT          ;TURN OFF THE RUN LIGHT.
7BF0 2695
7BF0 2696 00'00'21                                        LXI     H,POWER_BRANCH  ;RESTORE THE POWER FAIL VECTOR.
7BF3 2697 4C 25 11                                        LXI     D,POWER_VECTOR
```

```
7BF6 2698 0000'CD                                              CALL    MOVER_3
7BF9 2699
7BF9 2700
7BF9 2701
7BF9 2702                                       ;****************************************************************************
7BF9 2703                                       ;
7BF9 2704                                       ;
7BF9 2705                                       ;      TEST1C          DATA TEST ON STACK
7BF9 2706                                       ;      ------          ------------------
7BF9 2707                                       ;
7BF9 2708                                       ;              THE DATA PATH TO THE STACK WILL BE TESTED BY DOING A
7BF9 2709                                       ;      JSR AND RTS FOR EACH OF THE 15 ADDRESS BITS AND VERIFYING
7BF9 2710                                       ;      THE ADDRESS RETURNED FROM THE STACK.
7BF9 2711                                       ;
7BF9 2712                                       ;
7BF9 2713                                       ;      TEST STEPS:
7BF9 2714                                       ;
7BF9 2715                                       ;              1) LOAD UPC WITH TEST ADDRESS.
7BF9 2716                                       ;              2) PERFORM JSR.
7BF9 2717                                       ;              3) PERFORM RTS.
7BF9 2718                                       ;              4) COMPARE ADDRESS IN UPC FOP CORRECTNESS.
7BF9 2719                                       ;              5) PRINT ERROR IF ANY.
7BF9 2720                                       ;              6) REPEAT 1 - 5 FOR EACH TEST ADDRESS.
7BF9 2721                                       ;
7BF9 2722                                       ;
7BF9 2723                                       ;      ERROR1: DATA TEST OF STACK FAILURE.
7BF9 2724                                       ;
7BF9 2725                                       ;
7BF9 2726                                       ;      DEBUG:  1) CHECK ''BUS PC 0 - 14.
7BF9 2727                                       ;              2) CHECK ''PUSH STACK L'' ON THE JSR.
7BF9 2728                                       ;              3) CHECK ''ENABLE RTS L'' ON THE RTS.
7BF9 2729                                       ;              4) CHECK ''ENABLE SP L'' AND ''CSR 03 H'' ON THE.
7BF9 2730                                       ;                 STACK POINTER PAL.
7BF9 2731                                       ;
7BF9 2732                                       ;
7BF9 2733                                       ;****************************************************************************
7BF9 2734
7BF9 2735
7BF9 2736 0000'CD 1C 3E                         TEST1C:        HEAD    <^X1C>
          0F 0000'3A
          C9 0C06'D2
          0F 0000'3A
          3D D3 0C7F'DA
7C0F 2737
7C0F 2738 0055'2A                                              LHLD    ZERO_DAT       ;MOVE THE FIRST UPC ADDRESS TO BE
7C12 2739 006A'22                                              SHLD    TEMPA_DAT      ; WRITTEN INTO TEMPA_DAT IS ZERO.
7C15 2740
7C15 2741 C5 46 00'00'21     ********************              DO      15
          77 0F 3E           *
7C1D 2742                    *
7C1D 2743 006A'2A            *                                 LHLD    TEMPA_DAT      ;MOVE A DATA PATTERN TO EXP_DAT...
7C20 2744 0072'22            *                                 SHLD    EXP_DAT
7C23 2745                    *
7C23 2746 00'72'21           *                                 LXI     H,EXP_DAT      ;...AND INCREMENT IT.
7C26 2747 0000'CD            *                                 CALL    INC_WORD
7C29 2748                    *
7C29 2749 006A'2A            *                  1$:            LHLD    TEMPA_DAT      ;MOVE AN ADDRESS TO UPC_VALUE.
7C2C 2750 0000'22            *                                 SHLD    UPC_VALUE
7C2F 2751                    *
7C2F 2752 0000'CD            *                                 CALL    LOAD_UPC       ;LOAD THE ADDRESS INTO THE UPC.
```

```
7C32 2753                  *
7C32 2754 00'4C'21         *                          LXI     H,JSR_INST      ;JUMP TO A SUBROUTINE.
7C35 2755 01BA'CD          *                          CALL    EXEC_INST
7C38 2756                  *
7C38 2757 00'00'21         *                          LXI     H,NOP_INST      ;THIS CHECKS THAT SP IS NOT ALWAYS
7C3B 2758 01BA'CD          *                          CALL    EXEC_INST       ; ENABLED
7C3E 2759                  *
7C3E 2760 00'4F'21         *                          LXI     H,RET_INST      ;RETURN FROM THE SUBROUTINE.
7C41 2761 01BA'CD          *                          CALL    EXEC_INST
7C44 2762                  *
7C44 2763 0000'CD          *                          CALL    LOAD_UPC        ;MOVE THE UPC CONTENTS TO UPC_VALUE.
7C47 2764                  *
7C47 2765 00'00'21         *                          LXI     H,UPC_VALUE     ;MOVE THE UPC CONTENTS TO REC_DAT.
7C4A 2766 00'77'11         *                          LXI     D,REC_DAT
7C4D 2767 02 0E            *                          MVI     C,2
7C4F 2768 0000'CD          *                          CALL    MOVER
7C52 2769                  *
7C52 2770 00'72'21         * ****************         IFNEQ   EXP_DAT,REC_DAT,2        ;COMPARE THE EXPECTED AND
          02 0E 00'77'11   * *
          0C63'CA 0000'CD  * *
7C60 2771                  * *                                                       ; RECEIVED DATA.
7C60 2772                  * *
7C60 2773 035C'CD          * *                        CALL    TEST1C_ERROR    ;PRINT AN ERROR MESSAGE.
7C63 2774                  * *
7C63 2775                  * ****************         ENDIF
7C63 2776                  *
7C63 2777 0000'3A          *                          LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7C66 2778 0F               *                          RRC
7C67 2779 0C29'DA          *                          JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7C6A 2780                  *
7C6A 2781 37               *                          STC                     ;SET THE CARRY.
7C6B 2782                  *
7C6B 2783 00'69'21         *                          LXI     H,TEMPA_SHIFT   ;MAKE THE NEXT DATA PATTERN BY
7C6E 2784 0000'22          *                          SHLD    SHIFT_PNT       ; SHIFTING A ONE THROUGH THE
7C71 2785 0000'CD          *                          CALL    SHIFTER         ; PRESENT PATTERN.
7C74 2786                  *
7C74 2787                  *
7C74 2788 35 00'00'21      ******************         ENDDO
          70 C1 0C1D'C2
7C7D 2789
7C7D 2790 3C D3                                       TAIL
7C7F 2791
7C7F 2792
7C7F 2793                               ;*********************************************************************************
7C7F 2794                               ;
7C7F 2795                               ;
7C7F 2796                               ;       TEST1D          WCS STACK BIT 14 TEST
7C7F 2797                               ;       ------          ---------------------
7C7F 2798                               ;
7C7F 2799                               ;               THIS TEST WILL CONSIST OF EXECUTING THE SEQUENCE ''JSR RTS''
7C7F 2800                               ;       FIRST WITH A UPC VALUE OF ZERO AND CHECKING THAT UPC BIT 14 DOESN'T
7C7F 2801                               ;       GET SET, AND THEN WITH A UPC ADDRESS WITH BITS 0-13 SET. THE NEW
7C7F 2802                               ;       UPC VALUE SHOULD HAVE BIT 14 SET.
7C7F 2803                               ;
7C7F 2804                               ;
7C7F 2805                               ;       TEST STEPS:
7C7F 2806                               ;
7C7F 2807                               ;               1) LOAD UPC WITH ZERO.
7C7F 2808                               ;               2) EXECUTE ''JSR RTS''.
7C7F 2809                               ;               3) READ UPC TO SEE THAT BIT 14 IS CLEAR.
```

```
7C7F 2810                                 ;                4) PRINT ERROR IF ANY.
7C7F 2811                                 ;                5) LOAD UPC WITH ADDRESS THAT HAS BITS 0-13 SET.
7C7F 2812                                 ;                6) EXECUTE SEQUENCE ''JSR RTS''.
7C7F 2813                                 ;                7) READ UPC TO SEE THAT BIT 14 IS SET.
7C7F 2814                                 ;                8) PRINT ERRORS IF ANY.
7C7F 2815                                 ;                9) EXECUTE ''POP'' INSTRUCTION TO TEST NEXT STACK LOCATION.
7C7F 2816                                 ;               10) REPEAT 1-9 FOR EACH STACK LOCATION.
7C7F 2817                                 ;
7C7F 2818                                 ;
7C7F 2819                                 ;       ASSUMPTIONS:  THE OTHER BITS OF THE STACK HAVE ALREADY BEEN TESTED
7C7F 2820                                 ;                     BY MICRO CODE TESTS.
7C7F 2821                                 ;
7C7F 2822                                 ;
7C7F 2823                                 ;       ERROR1: WCS STACK BIT 14 TEST FAILURE.
7C7F 2824                                 ;
7C7F 2825                                 ;
7C7F 2826                                 ;
7C7F 2827                                 ;       DEBUG: CHECK HARDWARE STACK RAM
7C7F 2828                                 ;
7C7F 2829                                 ;
7C7F 2830                                 ;*****************************************************************************
7C7F 2831
7C7F 2832 0000'CD 1D 3E                   TEST1D:         HEAD    <^X1D>
          0F 0000'3A
          C9 0C8C'D2
          0F 0000'3A
          3D D3 0D23'DA
7C95 2833
7C95 2834 0055'2A                                         LHLD    ZERO_DAT        ;LOAD UPC ADDRESS WITH BIT 14
7C98 2835 0072'22                                         SHLD    EXP_DAT         ; CLEAR INTO EXP_DAT.
7C9B 2836
7C9B 2837 0055'2A                         1$:             LHLD    ZERO_DAT        ;LOAD UPC ADDRESS OF ZERO INTO
7C9E 2838 0000'22                                         SHLD    UPC_VALUE       ; UPC_VALUE.
7CA1 2839
7CA1 2840 0000'CD                                         CALL    LOAD_UPC        ;LOAD UPC ADDRESS INTO UPC.
7CA4 2841
7CA4 2842 00'4C'21                                        LXI     H,JSR_INST      ;PERFORM A JSR INSTRUCTION.
7CA7 2843 01BA'CD                                         CALL    EXEC_INST
7CAA 2844
7CAA 2845 00'4F'21                                        LXI     H,RET_INST      ;PERFORM A RETURN INSTRUCTION.
7CAD 2846 01BA'CD                                         CALL    EXEC_INST
7CB0 2847
7CB0 2848 0000'CD                                         CALL    LOAD_UPC        ;READ THE UPC.
7CB3 2849
7CB3 2850 0000'3A                                         LDA     UPC_VALUE       ;LOAD BIT 14 OF THE UPC INTO REC_DAT.
7CB6 2851 40 E6                                           ANI     <^X40>          ; ALL OTHER BITS ARE NOT OF INTEREST.
7CB8 2852 0077'32                                         STA     REC_DAT
7CBB 2853 00 3E                                           MVI     A,0
7CBD 2854 0078'32                                         STA     REC_DAT+1
7CC0 2855
7CC0 2856 00'72'21          ******************            IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'77'11    *
          0CD1'CA 0000'CD   *
7CCE 2857                   *                                                             ; RECEIVED DATA.
7CCE 2858                   *
7CCE 2859 037D'CD           *                             CALL    TEST1D_ERROR    ;PRINT AN ERROR MESSAGE.
7CD1 2860                   *
7CD1 2861                   ******************            ENDIF
7CD1 2862
7CD1 2863 0000'3A                                         LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
```

```
7CD4 2864 0F                                                   RRC
7CD5 2865 0C9B'DA                                              JC      1$               ;...JUMP TO BEGINNING OF ERROR LOOP.
7CD8 2866
7CD8 2867
7CD8 2868 0067'2A                                              LHLD    TEST1D_DAT+2     ;LOAD UPC ADDRESS WITH BITS 0-13 SET
7CDB 2869 0072'22                                              SHLD    EXP_DAT          ; INTO EXP_DAT.
7CDE 2870
7CDE 2871 0065'2A                              2$:             LHLD    TEST1D_DAT       ;LOAD UPC ADDRESS WITH BIT 14 SET
7CE1 2872 0000'22                                              SHLD    UPC_VALUE        ; INTO UPC_VALUE.
7CE4 2873
7CE4 2874 0000'CD                                              CALL    LOAD_UPC         ;LOAD UPC ADDRESS INTO UPC.
7CE7 2875
7CE7 2876 00'4C'21                                             LXI     H,JSR_INST       ;PERFORM A JSR INSTRUCTION.
7CEA 2877 01BA'CD                                              CALL    EXEC_INST
7CED 2878
7CED 2879 00'4F'21                                             LXI     H,RET_INST       ;PERFORM A RETURN INSTRUCTION.
7CF0 2880 01BA'CD                                              CALL    EXEC_INST
7CF3 2881
7CF3 2882 0000'CD                                              CALL    LOAD_UPC         ;READ THE UPC.
7CF6 2883
7CF6 2884 0000'3A                                              LDA     UPC_VALUE        ;LOAD BIT 14 OF THE UPC INTO REC_DAT.
7CF9 2885 40 E6                                                ANI     <^X40>           ; ALL OTHER BITS ARE NOT OF INTEREST.
7CFB 2886 0077'32                                              STA     REC_DAT
7CFE 2887 00 3E                                                MVI     A,0
7D00 2888 0078'32                                              STA     REC_DAT+1
7D03 2889
7D03 2890 00'72'21          ******************                 IFNEQ   EXP_DAT,REC_DAT,2        ;COMPARE THE EXPECTED AND
          02 0E 00'77'11    *
          0D14'CA 0000'CD   *
7D11 2891                   *                                                                   ; RECEIVED DATA.
7D11 2892                   *
7D11 2893 037D'CD           *                                  CALL    TEST1D_ERROR     ;PRINT AN ERROR MESSAGE.
7D14 2894                   *
7D14 2895                   ******************                 ENDIF
7D14 2896
7D14 2897 0000'3A                                              LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
7D17 2898 0F                                                   RRC
7D18 2899 0CDE'DA                                              JC      2$               ;...JUMP TO BEGINNING OF ERROR LOOP.
7D1B 2900
7D1B 2901 00'52'21                                             LXI     H,POP_INST       ;POP THE STACK TO TEST THE NEXT WCS
7D1E 2902 01BA'CD                                              CALL    EXEC_INST        ; LOCATION.
7D21 2903
7D21 2904 3C D3                                                TAIL
7D23 2905
7D23 2906                                      ;*********************************************************************************
7D23 2907                                      ;
7D23 2908                                      ;  LAST TEST FOR EVERY CONSOL_SECTION....FORCES END OF SECTION
7D23 2909                                      ;
7D23 2910                                      ;*********************************************************************************
7D23 2911
7D23 2912
7D23 2913 0000'CD 51 3E                        END_TEST:       HEAD    <^X51>
          0F 0000'3A
          C9 0D30'D2
          0F 0000'3A
          3D D3 0D3B'DA
7D39 2914
7D39 2915 3C D3                                                TAIL
7D3B 2916
```

```
$PSW                = 00000006              CSR_SHIFT             ********     X   02
$SP                 = 00000006              CSR_VALUE             ********     X   02
A                   = 00000007              D                   = 00000002
ALLOWP              = 00000003              DATA0                 ********     X   02
APTFLG              = 00000004              DATA1                 ********     X   02
AUTO                = 00000005              DATA2                 ********     X   02
B                   = 00000000              DATA3                 ********     X   02
BOOTEN              = 00000007              DATA_PNT              00000008 R       02
BOOTF               = 00000001              DATA_SHIFT            ********     X   02
C                   = 00000001              DCLO                = 00000002
CC_ADDR               ********     X   02   DISMEM              = 000000A7
CHECK_RD              0000007B R       02   DISPAR              = 000000A1
CLRACK              = 000000A9              DOLOOP                ********     X   02
CLRACL              = 00000033              E                   = 00000003
CLRATN              = 000000AB              ENBMEM              = 000000A6
CLRBSY              = 0000002E              ENBPAR              = 000000A0
CLRCLK              = 00000020              END_TEST              00000D23 R       02
CLRCSK              = 00000024              END_TEST19            00000ADD R       02
CLRCSR              = 00000038              END_TEST1B            00000BEE R       02
CLRDCL              = 00000031              ENTRY                 00000000 R       02
CLRHLT              = 000000AF              EOS_FLG               ********     X   02
CLRLIT              = 0000003C              ERROR_CON             ********     X   02
CLRMCI              = 00000029              ERROR_LOOP            00000B3A R       02
CLRMCK              = 0000002B              EXEC_INST             000001BA R       02
CLRPFI              = 000000AD              EXP_DAT               00000072 R       02
CLRSS               = 00000022              EXP_SHIFT             00000071 R       02
CLRTIM              = 000000A5              FAST_READ_WRO         0000016E R       02
CLRTMR              = 0000002C              FAST_WRITE_WRO        00000127 R       02
CLRUPC              = 000000A8              FILE_NAME             00000005 R       02
CLRUPK              = 00000026              GCVECT                ********     X   02
CLRWCK              = 00000036              H                   = 00000004
CLR_CPU_ACK           0000003A R       02   HEAD                = 00000000
CLR_CPU_ATTN          00000040 R       02   HEX_10              = 00000010
CLR_WRO_CC            0000002B R       02   HLTFLG              = 00000002
CNT                 = 0000081A              INC_WORD              ********     X   02
CNTA                = 00000FA0              INC_WRO               00000025 R       02
CNTLC_TYPED           ********     X   02   INC_WRO_CC            00000028 R       02
COMPARE               ********     X   02   INITH               = 00000035
COM_1A_1B_ERR         0000022C R       02   INITL               = 00000034
COM_1A_1B_ERR_1       00000227 R       02   ITCSR               = 0000001D
COM_ERR               00000218 R       02   ITDB                = 0000001C
COM_ERR_BYTE          00000206 R       02   JSR_INST              0000004C R       02
COM_ERR_BYTE_1        00000201 R       02   JUMP_WCS0             00000046 R       02
COM_ERR_OTHER         0000021D R       02   JUMP_WCS4             00000049 R       02
CON_ADD_DAT           ********     X   02   L                   = 00000005
CON_BEGIN_TEST        ********     X   02   LD_CSR_INST           000001C2 R       02
CON_ERROR             ********     X   02   LD_SUPPORT            0000008B R       02
CON_ERR_NUM           ********     X   02   LITERAL             = 00000001
CON_EXP_DAT           ********     X   02   LOAD_MOD_DAT          000001F8 R       02
CON_MOD_DAT           ********     X   02   LOAD_UPC              ********     X   02
CON_MSK_DAT           ********     X   02   LS_ADDR               00000007 R       02
CON_REC_DAT           ********     X   02   LS_READ               0000010A R       02
CPATTN              = 00000083              LS_WRITE              000000ED R       02
CPUACK              = 00000082              LVC                 = 00000000
CSR07               = 00000085              M                   = 00000006
CSR15               = 00000086              MACSTK1             = 0000081A
CSR23               = 00000087              MACSTK2             = 00000818
```

```
MACSTK3                          = 00000814                     SKIP_TEST                          *********    X    02
MCPU_A                             *********    X    02         SUPPORT_CODE                       0000000A R        02
MIPFLG                           = 000040D9                     SYM                              = 0000081A
MISC2                            = 00000001                     TEMPA_DAT                          0000006A R        02
MOVER                              *********    X    02         TEMPA_SHIFT                        00000069 R        02
MOVER_3                            *********    X    02         TEMPB_DAT                          0000006D R        02
MOVER_4                            *********    X    02         TEMPB_SHIFT                        0000006C R        02
MOV_LSOS_WR                        00000022 R        02         TEST10                             0000039E R        02
MOV_LSX_WRO                        0000001F R        02         TEST10_ERROR                       0000024D R        02
MOV_WRO_LSX                        0000001C R        02         TEST10_SHIFT_OS                    00000257 R        02
MWCS_A                             *********    X    02         TEST11                             00000439 R        02
NA_EXP_REC                         *********    X    02         TEST11_ERROR                       000002A7 R        02
NA_OTHER                           *********    X    02         TEST12                             00000500 R        02
NOP_CSR                            *********    X    02         TEST12_ERROR                       000002E0 R        02
NOP_INST                           *********    X    02         TEST13                             000005CF R        02
OKVT5                            = 00000007                     TEST13_ERROR                       000002EA R        02
ONE_DAT                            0000005A R        02         TEST14                             00000620 R        02
ONE_SHIFT                          00000056 R        02         TEST14_15_ERROR                    00000311 R        02
OS_ADD                             *********    X    02         TEST15                             000006A8 R        02
PARITY_JUMP                        *********    X    02         TEST16                             000007DC R        02
PARITY_VECTOR                    = 00004C22                     TEST16_ERROR                       0000031B R        02
POP_INST                           00000052 R        02         TEST17                             00000927 R        02
POWER_BRANCH                       *********    X    02         TEST17_18_ERROR                    00000325 R        02
POWER_VECTOR                     = 00004C25                     TEST18                             000009A2 R        02
READ                             = 00000080                     TEST19                             00000A1D R        02
READJ1                           = 00000003                     TEST19_DAT                         0000005F R        02
READJ2                           = 00000004                     TEST19_ERROR                       0000032F R        02
READ_BYTE_WRO                      0000018E R        02         TEST19_PAR_ERR                     00000A39 R        02
REC_DAT                            00000077 R        02         TEST1A                             00000ADE R        02
REC_SHIFT                          00000076 R        02         TEST1A_ERROR                       00000339 R        02
REMDIS                           = 00000006                     TEST1B                             00000B21 R        02
RET_INST                           0000004F R        02         TEST1B_2_3_ERROR                   00000352 R        02
SETACK                           = 000000A8                     TEST1B_DAT                         00000062 R        02
SETACL                           = 00000032                     TEST1B_ERROR                       00000343 R        02
SETATN                           = 000000AA                     TEST1B_TRAP                        00000B77 R        02
SETBSY                           = 0000002F                     TEST1C                             00000BF9 R        02
SETCLK                           = 00000021                     TEST1C_ERROR                       0000035C R        02
SETCSK                           = 00000025                     TEST1D                             00000C7F R        02
SETCSR                           = 00000039                     TEST1D_DAT                         00000065 R        02
SETDCL                           = 00000030                     TEST1D_ERROR                       0000037D R        02
SETHLT                           = 000000AE                     TTCR                             = 0000004B
SETLIT                           = 0000003D                     TTDB                             = 00000048
SETMCI                           = 00000028                     TTMODE                           = 0000004A
SETMCK                           = 0000002A                     TTSR                             = 00000049
SETPFI                           = 000000AC                     TUCR                             = 00000047
SETSS                            = 00000023                     TUDB                             = 00000044
SETTIM                           = 000000A4                     TUMODE                           = 00000046
SETTMR                           = 0000002D                     TUSR                             = 00000045
SETUPC                           = 000000A9                     UPC14                            = 00000084
SETUPK                           = 00000027                     UPC14A                           = 00000000
SETWCK                           = 00000037                     UPC_VALUE                          *********    X    02
SET_CPU_ACK                        0000003D R        02         VERSION                            00000003 R        02
SET_CPU_ATTN                       00000043 R        02         VNORML                           = 000000A3
SET_FOR_CONT                       *********    X    02         VSHIFT                           = 000000A2
SHIFTER                            *********    X    02         WCSB0                            = 00000008
SHIFTER1                           *********    X    02         WCSB1                            = 00000009
SHIFT_PNT                          *********    X    02         WCSB2                            = 0000000A
```

```
WCS_ADDRESS                          ********   X    02
WCS_BAD_PARITY                       ********   X    02
WCS_VALUE                            ********   X    02
WCS_WRITE                            ********   X    02
WRITE                              = 000000CC
WRITE_WCS                            ********   X    02
XD_ADDRS                             ********   X    02
XOR_WRO_LSO                          00000037 R      02
XOR_WRO_WR1                          00000034 R      02
X_CLR_WRO                            ********   X    02
X_CLR_WR1                            0000002E R      02
X_COM_WRO                            ********   X    02
X_COM_WR1                            00000031 R      02
X_MOV_WRWR                           ********   X    02
X_SHIFT_R                            ********   X    02
Y                                  = 00000050
YBUSRD                             = 000000EC
ZERO_DAT                             00000055 R      02
ZERO_SHIFT                           0000005B R      02

                                                   +-----------------+
                                                   ! Psect synopsis !
                                                   +-----------------+

PSECT name                          Allocation          PSECT No.  Attributes
----------                          ----------          ---------  ----------
.  ABS  .                           00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
. BLANK .                           00000000  (     0.)  01 (  1.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
CPU3                                00000D3B  ( 3387.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE

                                                +------------------------+
                                                ! Performance indicators !
                                                +------------------------+

Phase                        Page faults    CPU Time       Elapsed Time
-----                        -----------    --------       ------------
Initialization                        24    00:00:00.06    00:00:01.27
Command processing                    36    00:00:00.29    00:00:02.00
Pass 1                              1133    00:00:31.58    00:01:31.92
Symbol table sort                      2    00:00:00.42    00:00:01.73
Pass 2                              1309    00:00:12.52    00:00:33.74
Symbol table output                   30    00:00:00.18    00:00:00.38
Psect synopsis output                  2    00:00:00.02    00:00:00.08
Cross-reference output                 0    00:00:00.00    00:00:00.00
Assembler run totals                2542    00:00:45.10    00:02:11.14

The working set limit was 1000 pages.
338349 bytes (661 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 247 non-local and 103 local symbols.
4871 source lines were read in Pass 1, producing 47 object records in Pass 2.
148 pages of virtual memory were used to define 147 macros.
```

```
                                             +----------------------------+
                                             ! Macro library statistics !
                                             +----------------------------+
Macro library name                           Macros defined
------------------                           --------------
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1                      0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

/SHOW=(BIN)/LIST=ENKBC.LIS/OBJECT=ENKBC.OBJ 8085MAC.MAR+MACDEF.MAC+[]ENKBC.MAC
```